A Practical Method for Multilevel Classification and Accounting of Traffic in Computer Networks

Bujlow, Tomasz; Pedersen, Jens Myrup

Publication date:
2014

Document Version
Accepted author manuscript, peer reviewed version

A Practical Method for Multilevel Classification and Accounting of Traffic in Computer Networks

TOMASZ BUJLOW, JENS MYRUP PEDERSEN

Networking & Security
Department of Electronic Systems
Aalborg University

A Practical Method for Multilevel Classification and Accounting of Traffic in Computer Networks

Tomasz Bujlow and Jens Myrup Pedersen

Networking & Security
Department of Electronic Systems
Aalborg University

Tomasz Bujlow and Jens Myrup Pedersen. *A Practical Method for Multilevel Classification and Accounting of Traffic in Computer Networks.*

TECHNICAL REPORT

Version 1: February 4, 2014

Distribution:
Aalborg University
Department of Electronic Systems
Networking & Security
Fredrik Bajers Vej 7 A4
DK-9220 Aalborg
Denmark
Phone: +45 9940 8616
Fax: +45 9940 9840
netsec@es.aau.dk

DEN EUROPÆISKE UNION
Den Europæiske Fond for Regionaludvikling
Vi investerer i din fremtid

Abstract

Classification and accounting of computer network traffic is an important task of Internet Service Providers, as it allows for adjusting the bandwidth, the network policies, and providing better experience to their customers. However, existing tools for traffic classification are incapable of identifying the traffic in a consistent manner. The results are usually given on various levels for different flows. For some of them only the application is identified (as *HTTP*, *BitTorrent*, or *Skype*), for others only the content (as *audio*, *video*) or content container (as *Flash*), for yet others only the service provider (as *Facebook*, *YouTube*, or *Google*). Furthermore, Deep Packet Inspection (DPI), which seems to be the most accurate technique, in addition to the extensive needs for resources, often cannot be used by ISPs in their networks due to privacy or legal reasons. Techniques based on Machine Learning Algorithms (MLAs) require good quality training data, which are difficult to obtain. MLAs usually cannot properly deal with other types of traffic, than they are trained to work with – such traffic is identified as the most probable class, instead of being left unclassified. Another drawback of MLAs is their inability to detect the content carried by the flow, or the service provider.

To overcome the drawbacks of already existing methods, we developed a novel hybrid method to provide accurate identification of computer network traffic on six levels: *Ethernet*, *IP protocol*, *application*, *behavior*, *content*, and *service provider*. The *Ethernet* and *IP protocol* levels are identified directly based on the corresponding fields from the headers (*EtherType* in Ethernet frames and *Type* in IP packet). The *application* and *behavior* levels are assessed by a statistical classifier based on C5.0 Machine Learning Algorithm. Finally, *content* and *service provider* levels are identified based on IP addresses. The training data for the statistical classifier and the mappings between the different types of content and the IP addresses are created based on the data collected by Volunteer-Based System, while the mappings between the different service providers and the IP addresses are created based on the captured DNS replies. Support for the following applications is built into the system: America's Army, BitTorrent, DHCP, DNS, various file downloaders, eDonkey, FTP, HTTP, HTTPS, NETBIOS, NTP, RDP, RTMP, Skype, SSH, and Telnet. Within each application group we identify a number of behaviors – for example, for HTTP, we selected *file transfer*, *web browsing*, *web radio*, and *unknown*. Our system built based on the method provides also traffic accounting and it was tested on 2 datasets.

The classification results are as follows. On the *Ethernet* and *IP protocol* levels we achieved 0.00 % errors. The classification on the *application* and *behavior* levels were assessed together. Using the first dataset, we achieved 0.08 % of errors, while 0.54 % of flows remained as unknown. Using the second dataset, we achieved 0.09 % of errors, while 0.75 % of flows remained as unknown. Taking into account the *content* level, the classification using the first dataset gave us 0.22 % errors and 0.47 % of unclassified flows, while using the second dataset it gave us 0.96 % of errors and 1.42 % of unclassified flows. The classification on the *service provider* level was performed only using the first dataset (we needed the application-layer payloads) and it gave us 1.34 % of errors and 1.71 % of unknown flows. Therefore, we have shown that our system gives a consistent, accurate output on all the levels. We also showed that the results provided by our system on the application level outperformed the results obtained from the most commonly used DPI tools. Finally, our system was implemented in Java and released as an open-source project.

Acknowledgments

This research work was conducted in the context of a PhD project in the Section for Networking and Security in the Department of Electronic Systems at Aalborg University.

The project is co-financed by the European Regional Development Fund (ERDF)[1] and Bredbånd Nord A/S[2], a regional fiber networks provider. We are grateful, for the possibility to install and test our Volunteer-Based System, to *Gimnazjum nr 3 z Oddziałami Integracyjnymi i Dwujęzycznymi imienia Karola Wojtyły w Mysłowicach*[3], a high school in Poland, and to Bredbånd Nord A/S.

Finally, we are also very grateful to Josep Solé-Pareta, a professor from Universitat Politècnica de Catalunya (UPC) in Barcelona – the 3-months PhD stay at UPC in an excellent research environment significantly contributed to the knowledge used in creating this publication. The research in Spain was funded by the Spanish Ministry of Science and Innovation under contract TEC2011-27474 (NOMADS project) and by the Comissionat per a Universitats i Recerca del DIUE de la Generalitat de Catalunya (ref. 2009SGR-1140).

Aalborg, February 2014
Tomasz Bujlow and Jens Myrup Pedersen

[1] See http://ec.europa.eu/regional_policy/thefunds/regional/index_en.cfm
[2] See http://www.bredbaandnord.dk/
[3] See http://www.nr3.edu.pl/

About Authors

PhD Student: Tomasz Bujlow (tbu@es.aau.dk)
Supervisor: Jens Myrup Pedersen (jens@es.aau.dk)

Tomasz Bujlow is a Ph.D. Student in the Department of Electronic Systems at Aalborg University in Denmark. He received his Master of Science in Computer Engineering from Silesian University of Technology in Poland in 2008, specializing in Databases, Computer Networks and Computer Systems. Previously, he obtained his Bachelor of Computer Engineering from University of Southern Denmark in 2009, specializing in software engineering and system integration. His research interests include methods for traffic classification in computer networks. He is also a Cisco Certified Network Professional (CCNP) since 2010.

Jens Myrup Pedersen is working as an Associate Professor and the head of the Section for Networking and Security (NetSec) in the Department of Electronic Systems at Aalborg University. His current research interests include network planning, traffic monitoring, and network security. He obtained the degree of Master of Science in Mathematics and Computer Science from Aalborg University in 2002, and PhD degree in Electrical Engineering also from Aalborg University in 2005. He is an author/co-author of more than 80 publications in international conferences and journals, and has participated in Danish, Nordic and European funded research projects. He is also a board member of a number of companies within technology and innovation.

Contents

Abstract iii

Acknowledgments v

About Authors vii

Chapter 1 Introduction 1
1.1 The Background . . . 1
1.2 Existing Methods for Traffic Classification . . . 1
1.3 Our Contributions . . . 2
1.4 The Structure of the Report . . . 2

Chapter 2 Related Work 5
2.1 Classification Approaches Using MLAs . . . 5
2.2 Approaches for Obtaining Training Data . . . 5
2.3 Our Previous Approaches to Traffic Classification . . . 6
2.4 Traffic Identification by Clustering . . . 7

Chapter 3 Overview of the Methods 9
3.1 Classification Methods . . . 9
3.2 Training Methods . . . 10
3.3 Testing Methods . . . 10

Chapter 4 Design of the System 11
4.1 Traffic Capturing and Basic Classification Module . . . 11
4.2 Application and Behavior Classification Module . . . 11
4.3 Content Classification Module . . . 12
4.4 Service Provider Classification Module . . . 12
4.5 Traffic Accounting Module . . . 13
4.6 User Interface Module . . . 14

Chapter 5 Training of the System Modules 15
5.1 Application and Behavior Classification Module . . . 15
5.1.1 Obtaining the Training Dataset . . . 15
5.1.2 Labeling the Training Dataset . . . 17
5.1.3 Excluding the Problematic Flows . . . 18
5.1.4 Obtaining the Decision Tree . . . 18
5.2 Content Classification Module . . . 19
5.3 Service Provider Classification Module . . . 19
5.3.1 The Concept of Training . . . 19
5.3.2 The Training Solution . . . 20

Chapter 6 Evaluation of the System on a Dataset with Full Payloads **21**
6.1 Creating the Dataset . . . 21
6.2 Establishing the Ground Truth . . . 21
6.2.1 Ethernet Level . . . 21
6.2.2 IP Protocol Level . . . 21
6.2.3 Application and Behavior Levels . . . 22
6.2.4 Content Level . . . 22
6.2.5 Service Provider Level . . . 22
6.3 The Dataset . . . 22
6.4 Building and Testing of the Classifier . . . 25
6.4.1 Splitting of the Dataset . . . 25
6.4.2 Traffic Capturing and Basic Classification Module . . . 25
6.4.3 Application and Behavior Classification Module . . . 25
6.4.4 Content Classification Module . . . 25
6.4.5 Service Provider Classification Module . . . 26
6.5 Results . . . 26
6.5.1 Traffic Capturing and Basic Classification Module . . . 26
6.5.2 Application and Behavior Classification Module . . . 26
6.5.3 Content Classification Module . . . 27
6.5.4 Service Provider Classification Module . . . 27
6.6 Comparison with the Results from DPI Tools on the Application Level . . . 29

Chapter 7 Evaluation of the System on the Full VBS Dataset **31**
7.1 The Dataset . . . 31
7.2 Building and Testing of the Classifier . . . 33
7.3 Results . . . 33
7.3.1 Application and Behavior Classification Module . . . 33
7.3.2 Content Classification Module . . . 34

Chapter 8 Discussion and Comparison of the Results **35**

Chapter 9 Conclusions **39**

References **41**

Appendix A C5.0 Classification Attributes **47**

Appendix B Rules Used to Establish the Ground Truth on the *Application* and *Behavior* Levels **49**

Appendix C Rules Used to Detect the Suspicious Unknown Traffic **53**

Appendix D Service Providers Used During the Evaluation of the Classifier **55**

Chapter 1

Introduction

1.1 The Background

Classification and accounting of traffic in computer networks is an important task. To ensure the proper quality for the users, Internet Service Providers (ISPs) are required to know how their networks are being used. To satisfy their needs, information about the network usage must be presented on multiple levels, which characterize different aspects of the traffic: layer 3, layer 4 and application protocols, the behavior, the carried content, and the service provider. At first, the knowledge of how particular applications contribute to the traffic volume allows to adjust the network structure and settings. For example, users, who prefer to download large amounts of data using Peer-to-Peer (P2P) applications, can be offered higher bandwidth during the night, while the Quality of Service (QoS) policies in the network can be adjusted to support the most commonly used interactive applications, such as Skype. However, each application can be used in many different ways. For example, HTTP clients, as web browsers, can be used to browse pages, stream Internet radios or Internet TVs, or download big files. Therefore, to assure the proper quality of delivery, the network providers require the information about how the applications are being used. ISPs can buy bandwidth from multiple providers, which are characterized by different pricing and links of different quality to various service providers (as Yahoo, Google, Facebook, etc). To reduce the cost and provide better quality to the users, it is important to know which services are most commonly used and what kind of content (audio, video, etc.) is being offered by the services.

The first challenge would be where and how to monitor the traffic. The best places for traffic monitoring are access, distribution, or core links of the ISPs networks, as we cannot expect to install any software on all of the users' devices. The examination of the traffic must not violate the law and users' privacy. Moreover, the measurements must be done in real-time, or nearly real-time, to avoid storing huge amounts of data, which is not doable in high-speed infrastructures with links exceeding 10 Gbit/s, without involving large processing power and storage space.

1.2 Existing Methods for Traffic Classification

There are three main methods of traffic classification in computer networks: classification by transport-layer port numbers, Deep Packet Inspection (DPI), and statistical classification.

Distinguishing the traffic based on port numbers [1, 2] is a well-known method, which is implemented by many network devices, as routers and layer-3 switches. The method is fast, but it is not capable of detecting protocols which use dynamic port numbers, as Skype or other P2P [3–5]. The same concerns services, which use different port numbers than the default ones.

DPI is more flexible since it relies on inspection of the application payload [6]. Unfortunately, not all the applications leave patterns, which allow to precisely identify packets belonging to the applications. Therefore, some application protocols are not detected at all, overmatched (giving false positives), or undermatched (giving false negatives) – see the pattern descriptions of l-7 filter [7]. Classification methods, which use DPI, are slow and they require a lot of processing power, which makes them unfeasible for real-time processing in high-speed networks [3, 4]. Furthermore, it can be even impossible to use DPI if the payload is encrypted, the application signatures were changed [3], or the national law forbids to use this technique because of privacy and confidentiality concerns [3].

The statistical analysis, a newly emerged technology, is a reply to the drawbacks of the methods described above. There are two different techniques used in statistical identification of data: classification and clustering. Creating classification rules and data clusters based on statistical parameters by hand is slow and inefficient. For that purpose, tools based on Machine Learning Algorithms (MLAs) were invented, which are able to generate the rules or clusters based on sample data provided as the training input. The accuracy of statistical classification was assessed to be over 95 %, while the ability to distinguish various application protocols was comparable to DPI, and the speed was similar to port-based classification [1–3, 5, 6, 8–10]. However, current applications of MLAs have many drawbacks. At first, training of MLAs requires good quality data. Otherwise, we risk obtaining results of poor accuracy. At second, the classification attributes need to be carefully selected. If the classifier uses time-based attributes, the classification results can be biased by conditions in the network at the time of measurement. At third, if the classifier is trained to recognize only several selected types of traffic, its ability to classify traffic in the real network is also limited. In such case, any flow representing another type of traffic will be misclassified by being assigned to the most probable class of the included traffic types.

1.3 Our Contributions

All the classification methods described above have one thing in common: they are not able to identify the traffic on multiple levels in a consistent manner. The port-based classification usually gives the name of the application protocol. As we show later in the report, the results provided by the DPI tools are usually a mix of application names, content names, and service provider names, while no consistency on any level is preserved. Machine Learning based tools are not easily able to detect the content carried by the traffic or its service provider, so we developed a hybrid classification method, which accurately identifies the traffic on multiple levels: *Ethernet*, *IP protocol*, *application*, *behavior*, *content*, and *service provider*. Our system also provides traffic accounting, so operators are able to see the number of flows and the traffic volume on the selected classification levels, which passed the monitored network interface during the selected time. Moreover, the system has the ability to work also with the traffic, which is not defined to be identified by the system – the classification result is explicitly given as *UNKNOWN*, instead of assigning the flow to the most probable class, which is the most common behavior of similar tools. In this report, we show many different techniques (based on packet headers, C5.0 MLA, IP addresses) used to classify the traffic on multiple levels. The Machine Learning techniques use training data delivered by our host-based traffic monitoring tool, which ensures accuracy close to 100 % in data labeling. Furthermore, we present the full design, evaluation, and the practical open-source implementation of our system[1].

1.4 The Structure of the Report

The remainder of this report is structured as follows. Chapter 2 shows the related work in this area. Then, we start the description of our project by introducing the scientific methods in Chapter 3. The design of the

[1] Our Java implementation can be obtained for free from SourceForge [11]

particular modules of the system is described in Chapter 4. The process of obtaining the training data and creating the classification rules is shown in Chapter 5. Our system was evaluated on two sets of data; the first including full packet payloads (Chapter 6) and the second only packet headers (Chapter 7). The results were discussed and compared with the results obtained by other classification tools in Chapter 8. Chapter 9 concludes the report. Appendices at the end of the report contain supplementary information.

Chapter 2

Related Work

2.1 Classification Approaches Using MLAs

Usage of Machine Learning Algorithms (MLAs) in traffic classification is broadly covered by existing scientific literature. In [3], the authors succeeded in distinguishing 12 different applications by C4.5 based on first 5 packets in the flow with accuracy of around 97 %. J48 (a Java implementation of C4.5 with further improvements) was used in [4] to classify 5 different applications. The authors showed that it is possible to skip from 10 to 1000 packets from the beginning of a flow without significant decrease of the accuracy, which was fluctuating around 96 %. The attribute set used in this experiment included time-based attributes like flow duration and inter-arrival time. In [12], J48 was used to detect FTP and BitTorrent traffic based on size-dependent attributes, and the authors demonstrated that encryption mechanisms do not influence accuracy of the classification, which reached around 98 %.

In [2], the authors achieved 96-99 % of accuracy, while distinguishing traffic belonging to 5 different classes by C4.5. The classification attributes included those that require the possession of whole flows (like flow duration) as well as time-based attributes (as inter-arrival time distribution). It is worth mentioning that the training data were created by pre-classification based on ports. Flow duration as classification attribute was also used in [9], where the authors identified 6 applications with accuracy around 88 % using C4.5. The method described in [13] tries to recognize different kinds of HTTP traffic, including video progressive download, based on flows groups. This interesting Machine Learning approach uses Weka software. Another approach for traffic classification is described in [14] and is based on signatures contained by packets belonging to flows of particular types. Signature matching also requires processing of whole flows (at least until the signature is matched).

2.2 Approaches for Obtaining Training Data

The accuracy and results of traffic classification by MLAs are heavily dependent on the quality of the data based on which the classifier was trained. Inaccurate training data can result in many problems, as low classification accuracy, or high accuracy, but inappropriate classification of test cases (if training cases were pre-classified incorrectly). In all the papers referenced above, the training data were obtained in one of the following ways: collecting data from one application a time, port-based classification, DPI, statistical classification, or using public data traces. Drawbacks of most of the methods were already described. Collecting data from one application at a time, for example using Wireshark, is time-consuming, not scalable, and it requires a good separation of background traffic, such as DNS requests, or system updates.

Obtaining the ground truth can be based on already existing datasets. An example are Cooperative Association for Internet Data Analysis (CAIDA) data traces, which were collected in a passive or an active

way [15]. Another example is the Internet Measurement Data Catalog [16], also operated by CAIDA, which provides the references to different sources of data traces, which are available for research. The data are not stored by CAIDA itself, but on external servers [17]. Although the datasets are pre-classified (or they claim to contain only the traffic from the particular application / protocol), we do not know how the sets were created and how clean they are, which is a very important factor during testing traffic classifiers. MAWI repository [18] contains various packet traces, including daily 15-minutes traces made at a trans-Pacific line (150 Mbit/s link). The traces contain the first 96 bytes of the payload and they were used in several publications, including the ones about testing of different traffic classifiers [19]. However, their usefulness in testing of classifiers is quite limited since we do not know what they consist of. Another useful data source is the Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD) [20], which stores wireless trace data from many contributing locations. Some interesting comparison studies were made using datasets from different providers. In [21] the authors compare the data obtained from CAIDA and CERNET [22]. Many significant differences between them were found and they concern the lifetimes, lengths, rates of the flows, and the distribution of the TCP and UDP ports among them. Another interesting project is The Waikato Internet Traffic Storage (WITS) [23], which aims to collect and document all the Internet traces that the WAND Network Research Group from the University of Waikato has in their possession. Some of the traces can be freely downloaded and they contain traffic traces from various areas and of different types (as DSL residential traffic, university campus traffic, etc). Most of the traces do not have payload (it is zeroed or truncated).

2.3 Our Previous Approaches to Traffic Classification

To avoid problems with establishing of the ground truth for the use of the training data, at Aalborg University we constructed a new tool for collecting of network data, Volunteer-Based System (VBS), which relies on data collected from users, who agree to install the VBS client on their computers. The current architecture and implementation of VBS was presented in [24]. The system will be introduced in Section 5.1, as it was also used in our current approach for establishing the ground truth. We conducted several experiments and invented a few methods for traffic classification and Quality of Service (QoS) assessment using the data collected by VBS [25–28]. Other examples of usage of the VBS tool were obtaining statistics on the flow, application, and content levels [29, 30], and obtaining the ground truth for comparison of different DPI tools [31–33].

As the classification tool in all of our experiments, we chose C5.0, as this improved version of C4.5 is characterized by better performance and accuracy [34]. Our first approach involved recognizing traffic belonging to 7 different traffic groups: Skype, FTP, BitTorrent, web browsing, web radio, interactive gaming, and SSH. We achieved accuracy of over 99 % dependent on the classification options, by using only 35 packets from a random point in the flow [25]. However, in this experiment we did not sub-classify the traffic inside the *web browsing* class, which can be characterized by different behavior and which can carry various contents: regular website browsing, audio streaming, video streaming, file downloads, etc. Therefore, we updated VBS and included the possibility of inspecting the *content-type* header in HTTP packets. We found out that one transport-layer flow can contain multiple application-layer HTTP streams, which transmit various types of content: HTML files, web images, audio files, etc. The first packet (and only first packet) of each HTTP stream contains the *content-type* header, which indicates the type of the content. In [28], we demonstrated how to use this information in classification of HTTP traffic by C5.0. This initial approach had overall classification accuracy of around 85 %, but we observed poor accuracy between some traffic classes, e.g. 30 % of error between *video* and *file transfer*. Later we found out that this was caused by including in the *video* class not only the streamed content, but also video content, which was downloaded to the users' computers by HTTP (for example, from YouTube). Other concerns include the problem with precisely defining traffic behavior, as web browsing. The disability to deal with the unknown traffic (other than matched by our classes) was the last important issue, and the show stopper at the same time – without resolving this issue, the solution could not be implemented in the real network.

2.4 Traffic Identification by Clustering

Usage of MLAs in recognition of computer network traffic by clustering is less popular than classification. The most commonly used clustering algorithms are K-Means, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Expectation Maximization (EM), and other based on them [35–42]. As the environment for performing experiments, the scientist most often choose Waikato Environment for Knowledge Analysis (WEKA), which provides support for K-Means, COBWEB, DBSCAN, EM, Farthest First, Ordering Points to Identify the Clustering Structure (OPTICS), and many other clustering techniques. All these algorithms in the context of WEKA were described and compared in [43]. The authors have noticed that DBSCAN and EM algorithms do not require that the researchers know the number of clusters before the experiments starts (contrary to K-Means), and therefore, they are the most useful clustering tools to resolve real life problems. K-Means and DBSCAN were considered for TCP traffic classification of 8 different applications in [35]. Pre-classified public data traces (1000 and 2000 samples per category) were used as the input. The overall accuracy of clustering was assessed depending on the chosen data traces to be 79 % and 84 % for K-Means, while for DBSCAN it was 76 % and 72 %.

Another approach, which is able to detect both seen and unseen yet applications by Particle Swarm Optimization (PSO) algorithm was made in [36]. The authors found out that 5000 samples in the training data are sufficient to obtain the clustering accuracy of 7 different applications of around 95 %. The other papers, which describe the clustering techniques in traffic classification, also deal with quite low number of cases provided to the algorithm, for example, 5000 samples in total in [37] and 500 samples per application in [42]. The performance of the available clustering algorithms was not assessed while trying to construct the clusters from big amounts of data, which is necessary while trying to create an effective classifier for the core network traffic.

Chapter 3

Overview of the Methods

The methods presented in this paper can be divided into three groups used during traffic classification, training, and testing.

3.1 Classification Methods

Our system performs classification of Ethernet traffic independently on six levels:

1. Ethernet level (for example IP, IPX, APPLETALK)
2. IP protocol level (for example ICMP, TCP, UDP, EIGRP)
3. Application level (for example HTTP, SSH, BITTORRENT, EDONKEY)
4. Behavior level (for example WEBBROWSING, FILETRANSFER, STREAMING)
5. Content level (for example VIDEO, AUDIO)
6. Service provider level (for example YOUTUBE, MSN, WIKIPEDIA)

The traffic labeling is done by several modules, through which the traffic information is subsequently passed.

Traffic Capturing and Basic Classification Module. It is responsible for capturing all the traffic from the network and grouping it into flows based on different features, depending on the type of the traffic. The traffic is identified on *Ethernet* and *IP protocol* levels based directly on the *EtherType* field from Ethernet frames and the *Type* field from IP packets, so the results are accurate. Terminated flows (after 60 seconds of inactivity) are passed to the next modules.

Application and Behavior Classification Module. It is responsible for labeling the traffic on the *application* and *behavior* levels. Both levels are detected at once based on C5.0 Machine Learning Algorithm, which needs a file with the decision tree used during the classification process. By replacing the file containing the decision tree with a new version, we can add new applications and behaviors to the system.

Content Classification Module. It is based on two approaches. The first one uses static mappings between the former classification levels and the particular content types. The second one is based on dynamically generated mappings between IP addresses and the particular content types.

Service Provider Classification Module. It is based on two approaches. The first one uses static mappings between the former classification levels and the particular service provider names. The second one is based on dynamically generated mappings between the IP addresses and the particular service provider names.

3.2 Training Methods

Three modules of our system require an additional input before they are able to classify the traffic.

Application and Behavior Classification Module. The decision tree used by C5.0 is generated during the training process. The training cases are created based on the real network traces originated from our host-based monitoring software (see Section 5.1). The traces contain the process name associated with each flow. The process name is obtained from system sockets. We selected the applications and behaviors in order to cover the biggest amount of traffic in our traces collected from 54 users.

Content Classification Module. To establish the mappings between the IP addresses and the content types, we use the data collected by our host-based monitoring software. The collected information also contain values of the *content-type* field from HTTP headers associated with each HTTP-based flow. Based on this, we can find out, which types of content are the most commonly served by the particular IP addresses.

Service Provider Classification Module. The mappings between the IP addresses and the particular service provider names are obtained by another application created by us, which monitors DNS responses in the network. These contain the queried domain names and the associated IP addresses, making it possible to find out which service providers are the most commonly served by the particular IP addresses.

3.3 Testing Methods

We tested the system in two stages. At first, we used a dataset with full packet payloads, consisting of 1 262 022 flows (303 189 TCP and 958 833 UDP) and amounting for 35.69 GB (33.97 GB TCP and 1.72 GB UDP). It was created using a special version of our host-based monitoring software developed in collaboration with UPC (see Section 6.1) with help of 3 virtual machines used to generate various types of traffic. By using this dataset, we were able to test all modules of the classifier. Many modules of our system require training in advance, so our dataset was split into two disjoint sets; one was used for training, while the second was used for testing the classifier. Afterwards, the sets were swapped, so all flows were classified, while no flow was present in both training and testing dataset at the same time.

At second, we used a standard dataset created by our host-based monitoring software. This dataset did not include packet payloads and it was created through 1.5 years of monitoring of traffic from 54 users. It consisted of 23 333 721 flows (11 283 867 TCP and 12 049 854 UDP) amounting for 1 677.96 GB (1 114.85 GB TCP and 563.11 GB UDP). Using this dataset, we were able to evaluate the classification on the application, behavior, and content levels.

Chapter 4

Design of the System

4.1 Traffic Capturing and Basic Classification Module

This module is responsible for capturing all the Ethernet frames from the network and for the early stages of traffic classification on level 1 (*Ethernet*) and 2 (*IP protocol*). At first, the captured frames are inspected to obtain the value of the *EtherType* field, which determines the classification on level 1. The list of possible values for level 1 corresponds to the possible values of the *EtherType* field and can be found in the IEEE Public EtherType list [44]. Then, all the packets are grouped into flows in the following manner:

a) If the *EtherType* is IP, the network and transport layer headers are inspected as well. The flows are constructed based on the 5-tuple: local and remote IP addresses, local and remote ports, and the transport protocol name. As the *local* IP address, we consider the address, which is on the list of local IP addresses possessed by the system. Broadcast IP address is always recognized as the remote end. In case that neither the source nor the destination IP address is on the list of the local IP addresses, the local end is determined as the source of the first noticed packet of that flow. For each packet of the flow, we collect its size, direction, and state of the ACK and PSH flags. The level 2 class is determined based on the value of the *Type* field from the IP packet. The list of possible classes for level 2 can be found on the official IANA website [45]. If the module cannot determine the level 2 class, it marks it as *UNKNOWN*.

b) If the *EtherType* is of another type, the flows are constructed based on 3-tuple: local and remote MAC addresses, and the *EtherType* value. The local end is determined as the source of the first noticed packet of that flow, if neither the source or destination MAC address is the broadcast MAC, which is considered to be the remote end. For each packet of the flow, we collect only its size for accounting purposes. The level 2 class is always set to *UNKNOWN*.

The basic classification of flows is always accurate, since it is performed based directly on the values of fields from the Ethernet frame or IP packet. When no new packet in the flow is noticed for over 1 minute, the flow is considered to be terminated and it is provided to the next classification engine.

4.2 Application and Behavior Classification Module

This module is responsible for classification of the traffic on level 3 (*application*) and 4 (*behavior*). The classification is performed by C5.0, which is based on a machine learning algorithm, so the accuracy depends on how well the module is trained and which statistical parameters are taken into account. Only the traffic classified on level 2 as *TCP* or *UDP* needs to be examined by this module – the rest obtains the class

Table 4.1: List of Applications and Their Behaviors Recognized by Our System

Application	Behavior
AMERICASARMY	UNKNOWN, GAMESESSION
BITTORRENT	UNKNOWN, FILETRANSFER
DHCP	INTERACTIVE
DNS	INTERACTIVE
DOWNLOADER	UNKNOWN, FILETRANSFER
EDONKEY	UNKNOWN, FILETRANSFER
FTP	UNKNOWN, CONTROL, FILETRANSFER
HTTP	UNKNOWN, FILETRANSFER, WEBBROWSING, WEBRADIO
HTTPS	UNKNOWN
NETBIOS	UNKNOWN
NTP	INTERACTIVE
RDP	UNKNOWN
RTMP	STREAMING
SKYPE	UNKNOWN, CONVERSATION
SSH	UNKNOWN
TELNET	INTERACTIVE
UNKNOWN	UNKNOWN

UNKNOWN on both levels without any inspection. Both levels 3 and 4 are assessed together, since the behavior is directly associated with the particular application.

The list of possible values for levels 3 and 4 determined by this module is shown in Table 4.1. It was created to cover the most common types of traffic (regarding the number of flows and the number of transmitted bytes) collected by our user-based traffic analysis software and can be further extended when needed. The *UNKNOWN* behavior contains a mix of various application behaviors, which we are not able to separate and can also contain some elements, which belong to other, explicitly specified classes (as for example for *EDONKEY*, the *UNKNOWN* behavior level class can contain some elements of the *FILETRANSFER* class, which we were not able to separate). Note that the different behaviors can be evaluated only with respect to the particular application.

The classification relies on 50 attributes, which are described in Appendix A. We use mainly attributes based on packet sizes and to avoid using any time-related statistics, which are sensitive to disturbances in the network. The decision tree obtained from the training process allows to identify the application and behavior classes without the knowledge of anything besides the properties of the network and transport layers.

4.3 Content Classification Module

This module is responsible for classification of the traffic on level 5 (*content*). The classification relies either on static associations with the previous levels, or it is based on the IP addresses, where the module requires a file with mappings between the IP addresses and the types of the content. If there is no static association with the previous levels and if there are no mappings for the particular IP address, the content level class is assumed to be *UNKNOWN*. Therefore, the classification accuracy depends on the quality of the mappings. Table 4.2 shows the content classes assigned by the module to flows belonging to different level 3 and level 4 classes.

4.4 Service Provider Classification Module

This module is responsible for classification of the traffic on level 6 (*service provider*), which is based on several approaches:

Table 4.2: Mappings Between the Application & Behavior Classes and the Content

Application & Behavior	Content
HTTP FILETRANSFER	Based on IP addresses: AUDIO, VIDEO, UNKNOWN
HTTP WEBBROWSING	WWW
HTTP WEBRADIO	AUDIO
RTMP STREAMING	MULTIMEDIA
All the other classes	UNKNOWN

Table 4.3: List of Inconsistencies Between the Detected Levels

Application	Service
BITTORRENT	other than P2P
DHCP	other than LOCAL
EDONKEY	other than P2P
NETBIOS	other than LOCAL

a) At first, the class is tried to be identified based on the IP addresses. For that purpose, the classifier needs a list of IP addresses and the corresponding names of the service providers. Several different lists of IP addresses can be used based on the previous classifications of the flow (i.e., the flows previously classified as *HTTP FILETRANSFER VIDEO* can use another list than the flows classified as *RTMP STREAMING*, etc.). The classification accuracy depends on the quality of the mappings.

b) Yet unclassified flows are classified as *P2P* if they directly connect two end-users. It is assigned to flows, which application level was identified as: *BITTORRENT*, *EDONKEY*, or *SKYPE*.

c) Yet unclassified flows are classified as *LOCAL* if the remote IP address belongs to the private IP addresses pools, or if the application level associated with that flows is: *DHCP* or *NETBIOS*.

d) All other flows are classified as *UNKNOWN*.

Afterwards, this module verifies if the classifications on lower levels are consistent with the detected service. In case of misfits (i.e., a flow classified as *BITTORRENT* was estimated to originate from YouTube based on its IP), the classifications on the *application*, *behavior*, and *content* levels are being changed to *UNKNOWN*. The list of the inconsistent classifications is shown in Table 4.3.

4.5 Traffic Accounting Module

This module is responsible for accounting of the traffic. The information about the flows is directed to this module as the final step, after the flows are classified on all the levels. There are two modes available for logging: the basic mode, which creates aggregated statistics, and the advanced mode, which creates per-flow records. The following information is logged in the advanced mode:

- Timestamp of the start of the flow
- Local and remote MAC addresses (for non-IP flows only)
- Local and remote IP addresses (for IP flows only)
- Local and remote transport layer port numbers (for IP flows only)
- Number of packets and bytes in the flow, separately in both directions
- Results from the classification on all the six levels

In the basic mode, to conserve space, the results are stored aggregated by:

- The day of the measurement
- Local MAC address (for non-IP flows only)
- Local IP address (for IP flows only)
- Results from the classification on all the six levels

For each aggregate, we log the number of flows, packets, and Bytes, separately in both directions.

4.6 User Interface Module

The web-based user interface allows the authorized users to access the information collected in the database by the *Traffic accounting module*, providing the information about the number of flows and data volume transmitted by the particular users (or all users altogether) during the selected period of time (or from the beginning of the measurements). The information can be grouped or filtered according to all the classification levels.

Chapter 5

Training of the System Modules

Three of our system modules require an additional input, which is the result of adjusting the system to the current network environment:

a) *Application and behavior classification module* requires a file with decision trees, based on which the embedded C5.0 classifier can recognize the application and behavior associated with the particular flows.

b) *Content classification module* requires a file with mappings between the IP addresses and the most common types of content, which are from this IP address provided.

c) *Service provider classification module* requires a file with mappings between the IP addresses and the provider service names associated with them.

The process of obtaining the decision trees and the files with mappings needed for the particular modules is shown in Figure 5.1.

5.1 Application and Behavior Classification Module

5.1.1 Obtaining the Training Dataset

This module is based on C5.0 classifier, which uses a machine learning algorithm, so it must be trained on a properly labeled dataset. To create this dataset, we used Volunteer-Based System (VBS) for Research on the Internet developed at Aalborg University [24]. The tool was released under The *GNU General Public License v3.0* and it is published as a SourceForge project [11]. VBS consists of clients installed on machines belonging to volunteers and of the server with the central database. The clients collect the information about the flows of all Internet traffic data to and from the machine (i.e., start time of the flow, number of packets contained by the flow, local and remote IP addresses, local and remote ports, transport layer protocol) together with detailed information about each packet (i.e., direction, size, TCP flags, and relative timestamp to the previous packet). For each flow, the associated process name obtained from the system sockets is also collected. Additionally, the system collects some information about the types of transferred HTTP contents, for example *text/css* or *video/x-mpg*. The captured information is transmitted to the VBS server, which stores the data in a MySQL database. Therefore, the dataset created based on the information from VBS can be used as a source of ground truth for training of the C5.0 classifier. The design of VBS shown in Figure 5.2 was initially described in [46] and the current version with further improvements and refinements is shown in [24].

However, there are some limitations of VBS, which have an impact on the collected dataset. The socket must be open for at least 1 second to be noticed by VBS, so the process name could be collected. Therefore,

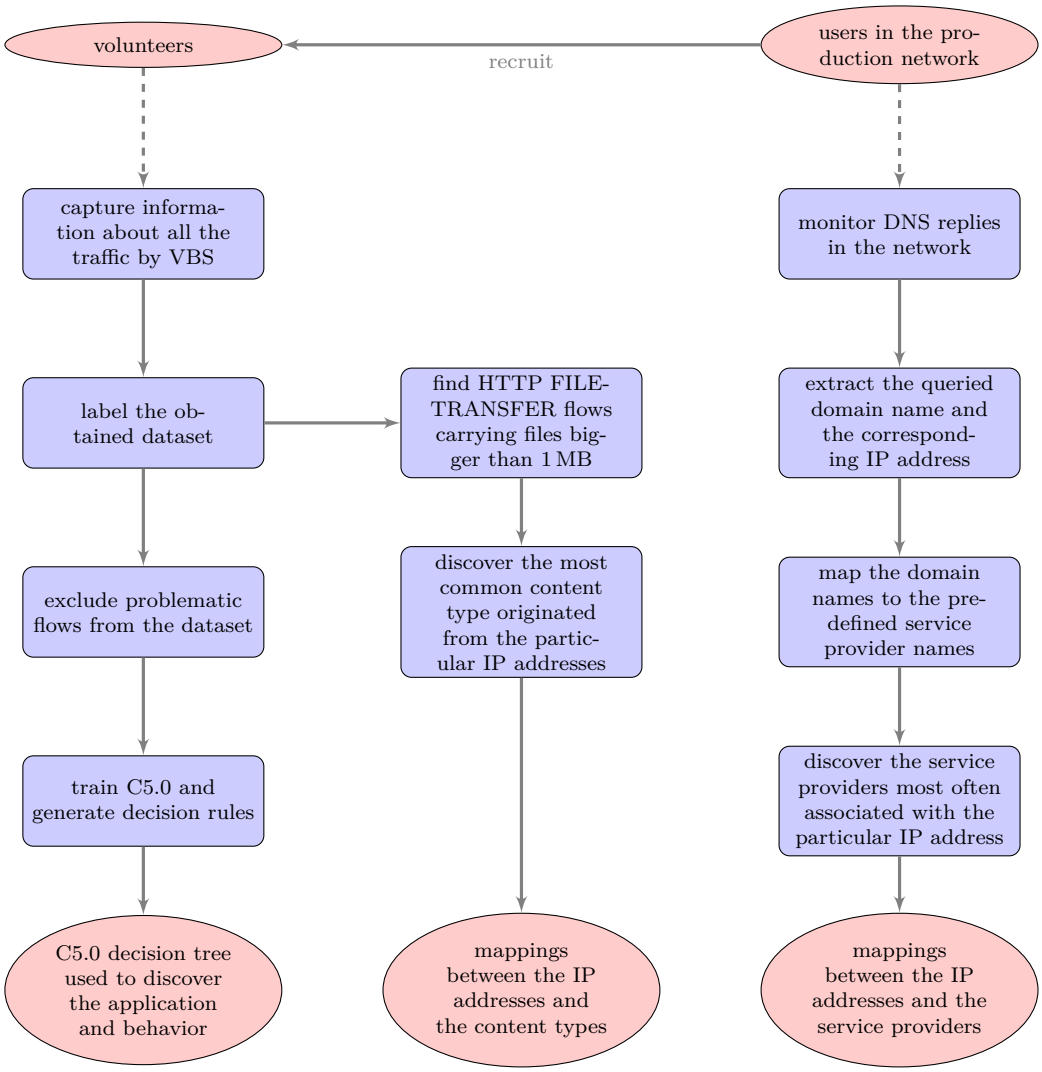

Figure 5.1: Training of the System Modules

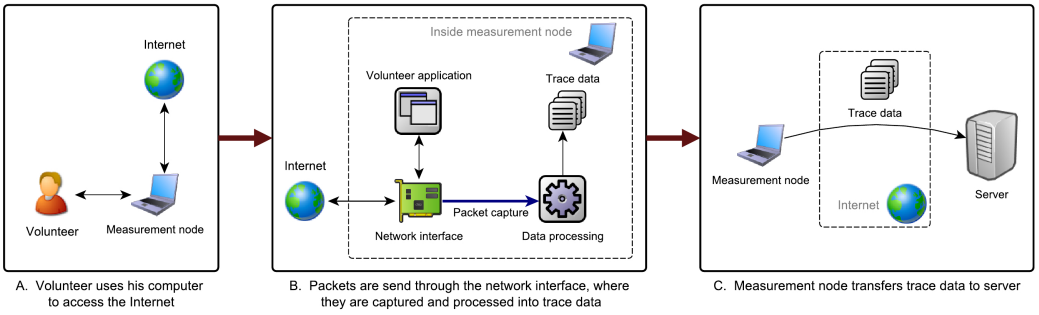

Figure 5.2: Overview of the VBS System [47]

in case of short flows (as 2-packets long DNS flows), the corresponding process name is not observed. Other limitations result from the host-monitoring nature of VBS, which collects only the data associated with the particular computer, on which it is installed. To create a representative dataset, the system must be installed on a representative group of users, who use their computers in a normal way.

5.1.2 Labeling the Training Dataset

At first, we needed to establish the ground truth by creating the rules, which the traffic needs to fulfill to be assigned to one of the *application* and *behavior* classes. All the rules were created by us manually, as it is not possible to automatically discover the character of the application based on the process name, or to discover the behavior of the traffic based on its general characteristics. The manual generation of classification rules is time-consuming, however, it allows us to make the classification consistent on all the levels. The rules are based mainly on our own observations regarding the traffic of the particular type. Another option would be to use an automatic training method, which could be based on results of DPI classification (as used in [48]), or on the process names from VBS. Unfortunately, by using output from DPI tools, we would mix the classifications on different levels. On the other hand, using the process names would create separate classes for each application (e.g., *uTorrent*, *BitTorrent*) instead of creating logical application groups. That would impose classification errors and add unneeded complexity, while the output would present lower value to the user. Additionally, there would be no way to discover the particular behavior of the application.

Labeling the flows with ground truth information on the *application* and *behavior* levels is split into two steps: we start by assigning the *application* class, and then, we assess the *behavior* within the particular application. The rules are processed from the most detailed to the most general, and only the previously unclassified cases are taken into account. For example, rules for *HTTP* traffic are processed before the rules for the *BITTORRENT* traffic, because BitTorrent clients also use HTTP to download updates, etc. Obviously, this traffic must be classified as *HTTP*, not as *BITTORRENT*. The rules used to establish the ground truth on the *application* and *behavior* levels for all the flows in the VBS database are shown in Appendix B.

Machine Learning based classification heavily depends on the quality of the training data, so the data used for training must be accurately assigned to the proper classes. However, in our case, the selected traffic classes are not disjoint but organized in supersets and subsets. For example, *HTTP WEBBROWSING* is a subset of *HTTP UNKNOWN*, which is in turn a subset of *UNKNOWN UNKNOWN* (see Figure 5.3). Subsets contain only these flows from the superset, which for sure belong to the particular subset. In many cases that imposes on us a big margin of tolerance, so that not all the flows from the superset are assigned to the subsets. We explain labeling of the HTTP traffic on the example given above. At first, all the flows, which contain at least one packet with an HTTP header, are moved from *UNKNOWN UNKNOWN* to *HTTP UNKNOWN* class. However, many flows were not captured from the beginning and they lack the first packets, which contain the HTTP header, for example, because the packet capturer crashed and was restarted. Therefore, some of the

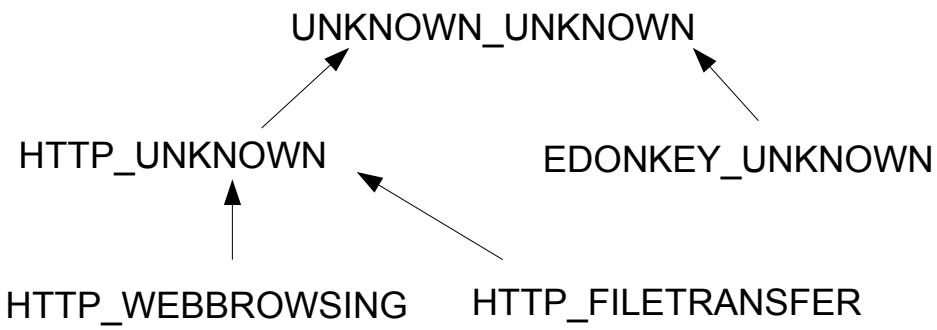

Figure 5.3: An example of Supersets and Subsets in Our Classification System

HTTP flows still remain in *UNKNOWN UNKNOWN* class. That is fully correct, however, the granularity of this labeling is lower than desired. Then, the flows, which fulfill the special criteria to be considered as interactive web flows, are moved from *HTTP UNKNOWN* to *HTTP WEBBROWSING* class. Again, the criteria to assess that the behavior is *web browsing* are based on the general characteristics of most flows of this type. Consequently, some flows representing this type of behavior will still remain in *HTTP UNKNOWN* class, which is correct, but the labeling granularity is lower than desired.

5.1.3 Excluding the Problematic Flows

As long as most of the flows from the superset are assigned to the subsets, Machine Learning based tools are able to use the data without decrease of accuracy. The classifier will still be properly trained to recognize the particular traffic classes, and during the classification on the test set, many flows from the superset will be classified to the proper subset (as many flows from *HTTP UNKNOWN* will be classified as *HTTP WEBBROWSING*). Therefore, the classification can generate more precise results than the granularity of the ground truth. However, in some cases the ground truth contains more elements of the particular class in the superset, instead of in the proper subset. That concerns short flows, as for example DNS flows, which are usually a few packets long, and for which we usually do not have captured the process names. Therefore, most of DNS flows remain in the *UNKNOWN UNKNOWN* class. If we use all the flows to generate the training data, we risk that some applications (as DNS) will not be classified at all.

To avoid this issue, we need to provide to the classifier only the training data, which allow to create accurate classification rules. The selection of the training data relies on excluding problematic flows, so that the subsets will always include more flows of the particular class than their supersets. Only the flows which do not have assigned application names and which do not contain any HTTP header are subject to be partly excluded. We established a threshold of 25 %, which means that the number of flows contained by the superset and suspicious to belong to its subset must not exceed 25 % of the number of flows contained by the subset. We are not going to exclude all the suspicious flows from the superset, since the superset can contain some flows which belong to other applications, and which match the same rules as we use during excluding these flows. The rules for detecting the suspicious traffic are described in Appendix C.

5.1.4 Obtaining the Decision Tree

Afterwards, from each flow contained by the training dataset we create a record of the training information for our C5.0 classifier, which was used by the the classifier to build the decision tree.

5.2 Content Classification Module

This module is based on two different approaches – static mappings between the *application* & *behavior* levels and the type of the content, and on the IP addresses. The static mappings were described before in Section 4.3, so we will focus on the mappings based on the IP addresses, which are the result of training. We use the mappings between the IP addresses and the type of the content only for the flows, which were classified as *HTTP FILETRANSFER*. To obtain the required mappings, we again decided to use the data collected by VBS, as it also collects the value of the content-type field in HTTP headers. At first, we examined all flows in the database, for which the ground truth was identified as *HTTP FILETRANSFER*. We searched inside them for files, which were at least 1 MB (as according to the rule, if a HTTP flow contains a file at least 1 MB, the whole flow is classified as *HTTP FILETRANSFER*) and compared the *content-type* field from their HTTP headers to the predefined list. If the file was of the type of *audio/xxx*, we extracted the source IP address of the file, and we assigned it to *AUDIO* class. If it was of the type of *video/xxx*, we extracted the source IP address of the file, and we assigned it to *VIDEO* class. If it was of another type, we extracted the source IP address of the file, and we assigned it to *UNKNOWN* class. After this step, we had records of IP addresses assigned to *AUDIO*, *VIDEO*, or *UNKNOWN* classes – one record for each transmitted file. The final mappings were obtained by finding the most common classes for the particular IP addresses. If the most common class for the particular IP address is *AUDIO*, we store this IP address and mark it as *AUDIO*. If the most common class for the particular IP address is *VIDEO*, we store this IP address and mark it as *VIDEO*. If the most common class for the particular IP address is *UNKNOWN*, we do not store this IP address at all.

5.3 Service Provider Classification Module

This module is also based on two different approaches – static mappings between the values of the *application* level and the service provider names, and the mappings between IP addresses and the service provider names created dynamically. The static mappings were described before in Section 4.4, and they are used after using the dynamic mappings, which have a higher priority. The dynamic mappings are the result of a training process.

5.3.1 The Concept of Training

Our concept assumes creating a database of IP addresses assigned to the most often used services, based on DNS responses. Due to privacy issues, VBS does not inspect DNS packets, so processing of the DNS responses in order to obtain the IP addresses associated with the particular services was moved to another project, which relies on network-based monitoring. On a computer, which has access to the network traffic, we installed our software, which inspects all DNS replies and extracts from them the name of the service, which was queried, together with all the IP addresses associated with that service name. If the questioned domain name was an alias (CNAME), all the DNS replies are recurrently processed and all the IP addresses are extracted. DPI of DNS replies is fast and lightweight, so a dedicated host is able to process all the DNS replies even in a high-speed network.

It is worth mentioning that one machine, which possesses a single IP address, can be used by many different network services offered by different providers. The same IP address can be used by, for example, both Google and Yahoo!, or the questioned domain name can correspond to a proxy. It is not useful to process the DNS responses directly by the main classifier and to extract the IP addresses in the real time. A DNS response can be buffered on a host for many days, when the buffered information can be used to get the IP address of the particular domain name. In the meantime, the host can obtain multiple DNS responses with other domain names corresponding to the same IP address, so there is no possibility to accurately obtain the real domain name associated with the particular flow by monitoring the packets in the network. It would be possible only by host-based monitoring, when all requests to DNS cache will be handled as well.

5.3.2 The Training Solution

There are two challenges in obtaining the list of mappings between the service provider names and the IP addresses:

- How to guess which service provider names should be included on the list? There are billions of different services existing in the Internet.
- How to assign a service name to a single IP, if we discovered that the same IP is used by different service providers?

The collected domain names, together with the IP addresses are stored in a temporary database table together with a counter, which informs us how many times the DNS responses contained the particular domain name associated with the particular IP address. Based on that, we can see which domains were questioned the most, and we can create appropriate rules for the service providers (as for example for YouTube, the domain names must contain *youtube.*, *ytimg.com*, or *youtube-nocookie.*). When a domain name matches the service rule, it is together with the corresponding IP addresses removed from the temporary table. Instead, a new, more general information is stored in the permanent database table: the IP address, the name of the service provider (as YouTube), and the occurrence rate. So the permanent table differs from the temporary one in two things: first, the service provider names are stored instead of the domain names. Second, aggregation is used, so all the domain names associated with the same service (as *youtube.*, *ytimg.com*, or *youtube-nocookie.* with YouTube) and the same IP address, are stored as a single entry.

Based on the permanent table, the final list with mappings between the service provider names and the IP addresses can be made. Normally, each IP address is associated with the service provider name, that amounts for the highest number of occurrences for the given IP address. However, we can have multiple different lists with mappings, which will be used depending on the previous classifications of the flow (i.e., flows previously classified as *HTTP FILETRANSFER VIDEO* can use another list than flows classified as *RTMP STREAMING*, etc.). This is useful when two or more services (such as YouTube and Doubleclik.net) use the same IP address, and we know, that Doubleclik.net does not serve videos, while YouTube does. For example, we can have a list used for flows identified as *HTTP FILETRANSFER VIDEO*. On the list, each IP address is associated with the name of the service provider known from providing video content, which amounts for the highest number of occurrences for the given IP address. If none of the listed for this IP address services is known to provide the video content, each IP address is associated with the service provider name, which amount for the highest number of occurrences for the given IP address (as in the normal case).

Chapter 6

Evaluation of the System on a Dataset with Full Payloads

The evaluation of the multi-level classifier requires a few steps. The dataset used for the evaluation is created, the ground truth is established by labeling the dataset on all the levels, and the dataset is divided into the training and the testing part. The former part is used for creating the necessary input files (decision trees, mappings between the IP addresses and the content types or service providers), and the latter part is used for the evaluation of the classification accuracy.

6.1 Creating the Dataset

To make any dataset useful for the purpose of the evaluation of our method, we needed to be able to label the traffic on all the levels. The dataset collected by VBS is able to deliver all the necessary information associated with each flow (as IP protocol, application name) but the name of the service provider. Therefore, we chose to use the dataset collected by a modified version of VBS, which we adjusted to collect also the packet payloads. It was used in collaboration with UPC BarcelonaTech to test the accuracy of different DPI tools (see the technical report published at UPC [31] and the conference paper [32]).

6.2 Establishing the Ground Truth

We started by establishing the ground truth – every flow from the dataset was classified on all the 6 levels using the objective information, including process names from the system sockets, and HTTP URLs from the HTTP headers.

6.2.1 Ethernet Level

All the flows on level 1 were classified as *IP*, since VBS collects only IP traffic. However, this did not influence the obtained classification accuracy – the classification on that level is also based on the EtherType field, so the classification results would always be the same as the ground truth.

6.2.2 IP Protocol Level

The flows were classified as *TCP* or *UDP* based on the information extracted from the packet headers. VBS collects only TCP and UDP traffic. However, this did not influence the obtained classification accuracy – the

classification on that level is also based on the IP protocol field, so the classification results would always be the same as the ground truth.

6.2.3 Application and Behavior Levels

To obtain the ground truth, we used the rules for generating the training data for the Machine-Learning based classifier on the application and behavior levels, which were processed in the same order as we described in Section 5.1. The rules were based on the process names from the system sockets and other objective information from VBS.

6.2.4 Content Level

To establish the ground truth, we made three steps. At first, we used the same static mappings as described in Section 4.3. Then, the class of *HTTP FILETRANSFER* flows (which is normally determined based on IP addresses), was determined based on the real content reflected by the *content-type* field from HTTP headers. Flows with the audio content got *AUDIO* class and flows with the video content got *VIDEO* class. Finally, all the yet unclassified flows were classified as *UNKNOWN*.

6.2.5 Service Provider Level

To establish the ground truth, we made the steps in the same order as described in Section 4.4. Although we had all the DNS packets together with their payload, we could not use them to generate the ground truth, as one IP address could correspond to many different service providers. Therefore, instead of using the DNS mappings, we classified the flows based on the URLs found in the HTTP header of the packets belonging to these flows. The limitation of this method is that we were able to classify only HTTP flows.

Based on the collected data, we chose 14 service providers to use in the experiment. The rules for matching the services to the flows are described in Appendix D. As the second step, we used the static mappings as described in Section 4.4. At the end, the flows, which were not assigned yet to any of these classes, became assigned to the *UNKNOWN* class. We are aware that during establishing the ground truth apart of the statically mapped flows, we were able to identify only HTTP flows, but this is the limitation of the information possessed by us. We expected that the created classifier would be able to identify correctly more flows, since it is based on the IP addresses, which makes it able to recognize also non-HTTP flows on the service provider level.

6.3 The Dataset

Based on the established ground truth, we show the statistics about the applications and their behaviors contained by our dataset. The dataset consisted of 1 262 022 flows (303 189 TCP and 958 833 UDP) amounting for 35.69 GB (33.97 GB TCP and 1.72 GB UDP).

We were not able to establish the ground truth on the application level for 59.25 % of flows, which amounted for 11.40 % of the total data volume. TCP flows amounted for 3.18 % of the total number of flows and 10.88 % of the total data volume without established ground truth. UDP flows amounted for 56.07 % of the total number of flows and 0.52 % of the total data volume without established ground truth. In fact, this is caused by a huge number of unclassified DNS flows, which are usually 2-packets long, so the corresponding socket application name could not be observed during the capture. The overall ground-truth classification of our dataset on all the levels is shown in Table 6.2.

For some HTTP flows, we detected the service provider based on the *URL* field in the HTTP header. The statistics for HTTP flows are shown in Table 6.1 – we labeled 64.35 % of HTTP flows amounting for 79.19 %

Table 6.1: Ground-Truth Classification of Service Providers for the HTTP Traffic

Service	**Flows**	**MB**	**Flows [%]**	**MB [%]**
YOUTUBE	2210	1949.19	3.55	30.06
UNKNOWN	22182	1349.45	35.65	20.81
ORACLE	103	841.74	0.17	12.98
FACEBOOK	5285	681.33	8.50	10.51
WIKIPEDIA	4078	508.19	6.55	7.84
YAHOO	16913	487.49	27.19	7.52
GOOGLE	3273	326.88	5.26	5.04
JUSTIN.TV	4075	134.05	6.55	2.07
UBUNTU.COM	247	102.65	0.40	1.58
MICROSOFT.COM	1301	55.56	2.09	0.86
DOUBLECLICK.NET	2072	38.72	3.33	0.60
TWITTER	297	7.59	0.48	0.12
TRIBALFUSION.COM	95	1.59	0.15	0.02
SCORECARDRESEARCH.COM	82	0.20	0.13	0.00

of the HTTP data volume. In total, for all the flows, we detected the service provider for 22.89 % of flows amounting for 28.97 % of the total data volume.

Table 6.2: Ground-Truth Classification of Our Dataset with Full Payloads

Ethernet	IP	Application	Behavior	Content	Service Provider	Flows	MB	Flows [%]	MB [%]
IP	TCP	BITTORRENT	FILETRANSFER	UNKNOWN	P2P	572	1046.94	0.05	2.86
IP	TCP	BITTORRENT	UNKNOWN	UNKNOWN	P2P	29967	75.94	2.37	0.21
IP	TCP	DNS	INTERACTIVE	UNKNOWN	UNKNOWN	2	0	0	0
IP	TCP	EDONKEY	FILETRANSFER	UNKNOWN	P2P	135	2755.48	0.01	7.54
IP	TCP	EDONKEY	UNKNOWN	UNKNOWN	P2P	1423	5.3	0.11	0.01
IP	TCP	FTP	CONTROL	UNKNOWN	UNKNOWN	61	0.58	0	0
IP	TCP	FTP	FILETRANSFER	UNKNOWN	UNKNOWN	797	3088.42	0.06	8.45
IP	TCP	FTP	UNKNOWN	UNKNOWN	UNKNOWN	18	0.06	0	0
IP	TCP	HTTP	FILETRANSFER	UNKNOWN	X	144	1631.6	0.01	4.46
IP	TCP	HTTP	FILETRANSFER	VIDEO	X	295	2160.21	0.02	5.91
IP	TCP	HTTP	UNKNOWN	UNKNOWN	X	15840	443.92	1.25	1.21
IP	TCP	HTTP	WEBBROWSING	WWW	X	45934	2248.9	3.63	6.15
IP	TCP	HTTPS	UNKNOWN	UNKNOWN	UNKNOWN	8539	833.85	0.68	2.28
IP	TCP	RDP	UNKNOWN	UNKNOWN	UNKNOWN	132907	13218.47	10.53	36.16
IP	TCP	RMTP	STREAMING	MULTIMEDIA	UNKNOWN	145	3209.49	0.01	8.78
IP	TCP	SSH	UNKNOWN	UNKNOWN	UNKNOWN	26219	91.8	2.08	0.25
IP	TCP	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	40191	3978.46	3.18	10.88
IP	UDP	BITTORRENT	FILETRANSFER	UNKNOWN	P2P	466	1433.98	0.04	3.92
IP	UDP	BITTORRENT	UNKNOWN	UNKNOWN	P2P	31840	64.51	2.52	0.18
IP	UDP	DNS	INTERACTIVE	UNKNOWN	UNKNOWN	6598	1.74	0.52	0
IP	UDP	EDONKEY	UNKNOWN	UNKNOWN	P2P	175023	63.1	13.87	0.17
IP	UDP	NETBIOS	UNKNOWN	UNKNOWN	LOCAL	9445	5.17	0.75	0.01
IP	UDP	NTP	INTERACTIVE	UNKNOWN	UNKNOWN	27786	4.03	2.2	0.01
IP	UDP	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	707675	188.95	56.07	0.52

6.4 Building and Testing of the Classifier

6.4.1 Splitting of the Dataset

We decided to split our dataset into 2 disjoint sets, where each of them contained approximately the same number of flows. Each flow was randomly assigned to either set 1 or set 2. Both sets were used during the training and testing of the classifier. At first, set 1 was used as the training set and test 2 as the test set, then the same procedure was done with both sets swapped. Thanks to that, we were able to test the classifier on all the flows contained by our dataset, while at the same time no flow was present both in the training and in the test set.

6.4.2 Traffic Capturing and Basic Classification Module

The evaluation was done on the real payloads of Ethernet frames contained by our dataset. The classification on level 1 (*Ethernet*) was based on the *EtherType* field from the Ethernet frame. It was sufficient to examine only the first packet in the flow to determine the carried protocol. Since VBS collects only IP traffic, we expected to obtain all the flows classified as *IP*.

In case, the obtained class was *IP*, the module tried to identify level 2 (*IP protocol*) class based on the information extracted from the packet headers (*IP protocol* field). As VBS collects only TCP and UDP traffic, we expected to obtain all the flows classified as *TCP* or *UDP*.

This module classified both sets (1 and 2) at once, since no training was needed on the levels on which the module operates.

6.4.3 Application and Behavior Classification Module

This module is based on C5.0 classifier, which uses a machine learning algorithm – the design and training methods were described in Section 4.2 and Section 5.1. Only the traffic classified on level 2 as TCP or UDP needed to be examined by this module. This module took at first set 1 as the training set and set 2 as the test set, then the sets were swapped. As we mentioned, the traffic classes in our dataset were not disjoint – they were organized as supersets, and subsets. In some cases (as for short DNS flows) the superset (as *UNKNOWN UNKNOWN*) could contain more items from the specific traffic class than the precise subset. In our case, according to *ipoque* PACE, the ground truth could be established only for 6 600 (0.97 %) DNS flows out of 678 381 DNS flows contained by our dataset, because the rest of the flows were collected without the associated application name. To avoid problems with proper training of the classifier, some data were excluded from the training sets. However, we did not exclude any data from the test sets – all the flows contained by our test sets were provided to the classifier unchanged. The method for excluding of the undesirable data from the training set was described in Section 5.1.

6.4.4 Content Classification Module

This module was based on two different approaches – static mappings between the application & behavior levels and the type of the content, and on the IP addresses. We used the same static mappings and the same method for obtaining the dynamic mappings based on IP addresses as described before in Section 5.2, but again we used our sets 1 and 2. The method was the same as we used previously, when the set 1 was at first used as the training set and set 2 as the test set, and then the training and test sets were swapped. The training set was used to generate the dynamic mappings between the IP addresses and the types of content, then the test set was used to evaluate the results.

6.4.5 Service Provider Classification Module

This module was built according to the method described in Section 5.3, which is the same as previously used, when set 1 was at first used as the training set and set 2 as the test set, and then the training and test sets were swapped.

The full packets from the training set were used as the input to the system, which was responsible for generating of the mappings between the IP addresses and the service provider names. The mappings were created using the same method as described in Section 5.3. Based on the services, which we had in our dataset, we decided to create 3 different mapping lists:

a) For the flows classified on the lower levels as *HTTP FILETRANSFER VIDEO*: the higher prioritized services were *YOUTUBE* and *FACEBOOK*

b) For the flows classified on the lower levels as *RTMP STREAMING*: the higher prioritized services were *JUSTIN.TV* and *YAHOO*

c) For the rest of the flows

Afterwards, we classified the traffic from the testing set using these mappings. Then, we applied all the static mappings based on the detected application names or IP addresses from the private pools. Finally, all the cases were checked for consistency with previous classifications and all the conflicts were solved by removing the faulty classifications as described in Section 4.4.

6.5 Results

The classification results are shown and discussed separately for each classification module.

6.5.1 Traffic Capturing and Basic Classification Module

As we expected, this module reached 0.00 % of error rate and classified properly all the cases. All 1 262 022 flows were classified on the *Ethernet* level as *IP*. On the *IP protocol* level, 303 189 (24.02 %) of the flows were classified as TCP and 958 833 (75.98 %) as UDP.

6.5.2 Application and Behavior Classification Module

The results obtained by this module are shown in Table 6.3. We present the number of flows in each traffic category (based on the ground truth) together with the percentage of flows, which were classified correctly, wrong, or as *UNKNOWN*. There are two categories for the flows correctly classified. The first category, *correct*, means that the flows were assigned to the same class as the original one or to a subset of the original class. For example, flows from the *BITTORRENT UNKNOWN* class were classified as *BITTORRENT UNKNOWN*, or *BITTORRENT FILETRANSFER*. The second category, *correct-lg*, means that we obtained results of lower granularity than the original class, but still both application and behavior levels are not misclassified. For example, flows from the *BITTORRENT FILETRANSFER* class were classified as *BITTORRENT UNKNOWN*.

As observed, the total error rate for all the flows was 0.08 %, while 0.54 % of flows remained unknown. If we look at the average from all the traffic classes, the error rate was slightly bigger (0.71 %) and 5.86 % of flows remained unknown. Additionally, we tried to identify 747 866 flows, which ground truth was given as *UNKNOWN UNKNOWN*: 88.49 % of them were classified as *DNS INTERACTIVE*, 4.20 % as *EDONKEY UNKNOWN*, 1.13 % as other classes, while only 6.18 % remained as *UNKNOWN UNKNOWN*. It means that the classifier built by us was able to recognize almost all the traffic for which we were not able to establish the ground truth due to the unknown application name.

Table 6.3: Classification by the *Application and Behavior Classification Module* for the Particular Types of the Traffic

Application & Behavior	Flows	Correct [%]	Correct-LG [%]	Wrong [%]	Unknown [%]
BITTORRENT FILETRANSFER	1038	97.59	0.39	1.16	1.06
BITTORRENT UNKNOWN	61807	98.17	0.00	0.22	1.61
DNS INTERACTIVE	6600	95.00	0.00	0.03	4.97
EDONKEY FILETRANSFER	135	87.41	2.22	4.44	5.93
EDONKEY UNKNOWN	176446	99.52	0.00	0.08	0.40
FTP CONTROL	61	98.36	0.00	0.00	1.64
FTP FILETRANSFER	797	97.24	0.00	1.00	1.76
FTP UNKNOWN	18	33.33	0.00	0.00	66.67
HTTP FILETRANSFER	439	92.94	4.78	1.82	0.46
HTTP UNKNOWN	15840	99.72	0.00	0.21	0.06
HTTP WEBBROWSING	45934	87.38	12.47	0.10	0.05
HTTPS UNKNOWN	8539	93.64	0.00	0.20	6.16
NETBIOS UNKNOWN	9445	100.00	0.00	0.00	0.00
NTP INTERACTIVE	27786	100.00	0.00	0.00	0.00
RDP UNKNOWN	132907	100.00	0.00	0.00	0.00
RTMP STREAMING	145	88.97	0.00	2.76	8.28
SSH UNKNOWN	26219	99.49	0.00	0.02	0.50
For all flows together	514156	98.26	1.12	0.08	0.54
Average from all classes		92.28	1.17	0.71	5.86

Table 6.4: Classification by the *Content Classification Module*

Content	Flows	Correct [%]	Wrong [%]	Unknown [%]
MULTIMEDIA	145	88.97	0.00	11.03
VIDEO	295	77.29	0.00	22.71
WWW	45934	87.38	0.00	12.62
UNKNOWN	1215648	99.78	0.22	0.00
All flows	1262022	99.32	0.22	0.47
Average from classes		88.36	0.06	11.59

6.5.3 Content Classification Module

The accuracy of the classification of the particular types of contents is shown in Table 6.4. As presented, we did not obtain any false classification within the flows from the *MULTIMEDIA*, *VIDEO*, and *WWW* classes. The rate of the unclassified flows is heavily influenced by the size of the training and testing sets used in the experiment. In our case, the ground truth for the flows labeled as *UNKNOWN* was established as non-*MULTIMEDIA*, non-*VIDEO*, and non-*WWW*. We obtained 0.22 % of errors for this class, which means that the same IP addresses were used both for the flows transmitting files from the *MULTIMEDIA*, *VIDEO*, and *WWW* classes, and for flows transmitting other kinds of files. In total, we classified properly 99.32 % of flows, 0.22 % of flows were classified wrong, and 0.47 % remained unknown.

6.5.4 Service Provider Classification Module

The accuracy of classification of particular service providers is shown in Table 6.5. The correct service provider name was given to 96.95 % of classified flows, the wrong service provider name was given to 1.34 % of flows, while for 1.71 % it remained unknown. A significant percent of errors originated from *DOUBLECLICK.NET* and *SCORECARDRESEARCH.COM*, which represent advertising services. It looks like their physical infrastructure is not centralized, but distributed among servers of other services, to which they supply advertisements. We also observed a significant misclassification between *GOOGLE* and *YOUTUBE*, as it seems that they use the same IP addresses.

Table 6.5: Classification by the *Service Provider Classification Module*

Service provider	**Flows**	**Correct [%]**	**Wrong [%]**	**Unknown [%]**
DOUBLECLICK.NET	2072	70.70	26.59	2.70
FACEBOOK	5285	97.35	1.06	1.59
GOOGLE	3273	41.34	42.65	16.01
JUSTIN.TV	4075	50.38	8.20	41.42
LOCAL	9445	100.00	0.00	0.00
MICROSOFT.COM	1301	84.24	15.14	0.61
ORACLE	103	65.05	19.42	15.53
P2P	239426	99.25	0.01	0.74
SCORECARDRESEARCH	82	1.22	95.12	3.66
TRIBALFUSION.COM	95	85.26	0.00	14.74
TWITTER	297	85.19	7.07	7.74
UBUNTU.COM	247	99.19	0.00	0.81
WIKIPEDIA	4078	98.58	0.81	0.61
YAHOO	16913	91.12	4.88	4.00
YOUTUBE	2210	82.67	15.02	2.31
All flows	288902	96.95	1.34	1.71
Average from classes		76.77	15.73	7.50

Table 6.6: DPI Tools Included in Our Comparison

Name	Version	Released	Identified Applications
PACE	1.41	June 2012	1000
OpenDPI	1.3.0	June 2011	100
nDPI	rev. 6391	March 2013	170
L7-filter	2009.05.28	May 2009	110
Libprotoident	2.0.6	Nov 2012	250
NBAR	15.2(4)M2	Nov 2012	85

6.6 Comparison with the Results from DPI Tools on the Application Level

The dataset containing packets with full payloads was used previously in [31] and [32] to evaluate the accuracy of various DPI traffic classifiers: PACE, OpenDPI, NDPI, Libprotoident, NBAR, and two different variants of L7-filter. Therefore, all the flows contained by the dataset were labeled by all of the DPIs, which allowed us to compare their accuracy with the accuracy of our system. Table 6.6 shows the versions of the DPI tools used in the experiment together with the number of applications recognized by them. L7-filter was tested in two different configurations. In the first version (*L7-filter-all*), we activated all the patterns, giving a low priority to the patterns marked as *overmatching*. In the second version (*L7-filter-sel*), the patterns declared as *overmatching* were not activated. In both cases, the patterns are matched to the first 10 packets or the first 10 000 B in each traffic direction.

All the tools provide results on different levels of granularity. For example, Libprotoident provides only transport protocol name if it is not able to detect the application. On the other hand, PACE, OpenDPI and NDPI sometimes provide only the content-level classification (as *FLASH*) without indicating what the application really is (for *FLASH* it can be *HTTP*, *RTMP*, etc). Because most of the results obtained from these DPI tools are on the *application* level, we decided to compare the classification results on our *application* level. The results provided by the DPI tools were considered to be correct only if the proper result on the *application* level was returned. In the comparison, we included only these cases, for which the ground truth was obtained. The percent of correct classification by all the tools is shown in Table 6.7. The comparison shows that our multilevel classification system provided very precise results. The overall accuracy of our system was 99.38 % and the average per-class accuracy was 97.44 %. The values are higher than the accuracy of the tested DPI tools.

Table 6.7: Comparison of % Correct Classifications for the Particular Applications by Our System and Various DPI Tools

Application	Flows	Multi-level	PACE	Open-DPI	NDPI	Libproto-ident	NBAR	L7-Filter-All	L7-Filter-Sel
BITTORRENT	62845	98.14	81.40	27.09	55.92	77.19	27.30	42.22	42.24
DNS	6600	95.00	99.97	99.97	99.88	99.97	99.97	98.95	98.95
EDONKEY	176581	99.51	94.80	0.45	0.45	98.40	0.38	34.22	0.00
FTP	876	95.32	96.00	95.32	95.32	94.63	39.27	5.14	5.14
HTTP	62213	99.83	92.50	94.00	73.89	98.88	99.35	4.41	27.84
HTTPS	8539	93.64	91.73	93.76	39.67	92.65	85.62	72.99	90.06
NETBIOS	9445	100.00	66.66	24.63	100.00	0.00	100.00	0.00	0.00
NTP	27786	100.00	100.00	100.00	100.00	100.00	0.40	99.83	0.00
RDP	132907	100.00	99.06	99.09	99.07	98.85	0.00	0.00	0.00
RTMP	145	88.97	0.00	0.00	0.00	87.59	0.00	0.00	0.00
SSH	26219	99.49	95.57	95.59	95.59	95.71	99.24	95.71	95.71
For all flows together	514156	99.38	93.78	54.19	55.76	94.04	25.18	30.21	16.19
Average from all classes		97.44	84.29	65.34	67.96	86.49	48.06	40.31	31.34

Chapter 7

Evaluation of the System on the Full VBS Dataset

After the small-scale system evaluation on the dataset containing full packet payloads, we decided to repeat the experiment on a much bigger dataset. For this purpose, we used the full dataset collected by VBS and stored at Aalborg University. The dataset contained traces collected by VBS clients installed on 54 machines belonging to volunteers in Denmark and Poland during around 1.5 years from December 27, 2011 to June 12, 2013. The volunteers were both private and corporate users. Such kind of setup ensures high diversity of the data despite the relatively limited number of installations. VBS in its original version did not collect packet payloads, so we were able to test the classification only on the *application*, *behavior*, and *content* levels. However, the classification results on the *Ethernet* and *IP provider* levels are accurate by nature, so no testing was needed. So, the results of the classification on the *service* level were the only ones, of which accuracy we were not able to assess for that dataset. The method of obtaining the ground truth for the *application*, *behavior*, and *content* levels was the same as for the dataset with full packet payloads, which was described previously in Section 6.2.

7.1 The Dataset

Based on the established ground truth, we show the statistics about the applications and their behaviors contained by our dataset. The dataset consisted of 23 333 721 flows (11 283 867 TCP and 12 049 854 UDP) amounting for 1 677.96 GB (1 114.85 GB TCP and 563.11 GB UDP).

We were not able to establish the ground truth on the *application* level for 26.33 % of flows, which amounted for 15.33 % of the total data volume. TCP flows amounted for 13.08 % of the total number of flows and 10.52 % of the total data volume without established ground truth. UDP flows amounted for 13.25 % of the total number of flows and 4.81 % of the total data volume without established ground truth. In fact, this was caused by a huge number of unclassified DNS flows, which are usually 2-packets long, so the corresponding socket application name could not be observed during the capture. The overall ground-truth classification of our VBS dataset is shown in Table 7.1.

Table 7.1: Ground-Truth Classification of Our VBS Dataset

IP Protocol	Application	Behavior	Content	Flows	MB	Flows [%]	MB [%]
TCP	BITTORRENT	FILETRANSFER	UNKNOWN	32774	154287.09	0.14	8.98
TCP	BITTORRENT	UNKNOWN	UNKNOWN	3235831	35263.55	13.87	2.05
TCP	DNS	INTERACTIVE	UNKNOWN	5	0.01	0.00	0.00
TCP	DOWNLOADER	FILETRANSFER	UNKNOWN	284	20499.20	0.00	1.19
TCP	DOWNLOADER	UNKNOWN	UNKNOWN	122503	3421.99	0.53	0.20
TCP	EDONKEY	FILETRANSFER	UNKNOWN	2719	71881.22	0.01	4.18
TCP	EDONKEY	UNKNOWN	UNKNOWN	22700	1809.43	0.10	0.11
TCP	FTP	CONTROL	UNKNOWN	105	0.35	0.00	0.00
TCP	FTP	FILETRANSFER	UNKNOWN	124	11713.73	0.00	0.68
TCP	FTP	UNKNOWN	UNKNOWN	11	0.06	0.00	0.00
TCP	HTTP	FILETRANSFER	AUDIO	1439	9313.63	0.01	0.54
TCP	HTTP	FILETRANSFER	UNKNOWN	9726	212182.14	0.04	12.35
TCP	HTTP	FILETRANSFER	VIDEO	23511	189091.28	0.10	11.01
TCP	HTTP	UNKNOWN	UNKNOWN	1583731	50895.30	6.79	2.96
TCP	HTTP	WEBBROWSING	WWW	2355298	72690.14	10.09	4.23
TCP	HTTP	WEBRADIO	AUDIO	326	19012.81	0.00	1.11
TCP	HTTPS	UNKNOWN	UNKNOWN	748695	37507.37	3.21	2.18
TCP	RDP	UNKNOWN	UNKNOWN	12	0.01	0.00	0.00
TCP	RTMP	STREAMING	MULTIMEDIA	497	67996.21	0.00	3.96
TCP	SKYPE	CONVERSATION	UNKNOWN	1588	2925.00	0.01	0.17
TCP	SKYPE	UNKNOWN	UNKNOWN	89557	346.54	0.38	0.02
TCP	UNKNOWN	UNKNOWN	UNKNOWN	3052431	180764.30	13.08	10.52
UDP	AMERICASARMY	GAMESESSION	UNKNOWN	80	111.36	0.00	0.01
UDP	AMERICASARMY	UNKNOWN	UNKNOWN	6260	1.69	0.03	0.00
UDP	BITTORRENT	FILETRANSFER	UNKNOWN	52931	391440.00	0.23	22.78
UDP	BITTORRENT	UNKNOWN	UNKNOWN	8200616	65932.23	35.14	3.84
UDP	DHCP	INTERACTIVE	UNKNOWN	4459	2.82	0.02	0.00
UDP	DNS	INTERACTIVE	UNKNOWN	158314	47.07	0.68	0.00
UDP	DOWNLOADER	UNKNOWN	UNKNOWN	9872	18.44	0.04	0.00
UDP	EDONKEY	UNKNOWN	UNKNOWN	59896	19.18	0.26	0.00
UDP	NETBIOS	UNKNOWN	UNKNOWN	13133	6.45	0.06	0.00
UDP	NTP	INTERACTIVE	UNKNOWN	58375	8.57	0.25	0.00
UDP	SKYPE	CONVERSATION	UNKNOWN	892	36256.00	0.00	2.11
UDP	SKYPE	UNKNOWN	UNKNOWN	394005	163.82	1.69	0.01
UDP	UNKNOWN	UNKNOWN	UNKNOWN	3091021	82611.96	13.25	4.81

Table 7.2: Classification by the *Application and Behavior Classification Module* for the Particular Types of the Traffic in VBS Dataset

Application & Behavior	Flows	Correct [%]	Correct-LG [%]	Wrong [%]	Unknown [%]
AMERICASARMY GAMESESSION	80	78.75	0.00	15.00	6.25
AMERICASARMY UNKNOWN	6260	99.89	0.00	0.08	0.03
BITTORRENT FILETRANSFER	85705	99.47	0.11	0.12	0.30
BITTORRENT UNKNOWN	11436447	99.78	0.00	0.02	0.20
DHCP INTERACTIVE	4459	99.57	0.00	0.00	0.43
DNS INTERACTIVE	158319	91.05	0.00	0.08	8.88
DOWNLOADER FILETRANSFER	284	70.42	8.10	15.14	6.34
DOWNLOADER UNKNOWN	132375	96.13	0.00	2.06	1.81
EDONKEY FILETRANSFER	2719	92.42	0.88	4.74	1.95
EDONKEY UNKNOWN	82596	96.53	0.00	1.99	1.48
FTP CONTROL	105	79.05	0.00	0.00	20.95
FTP FILETRANSFER	124	71.77	0.00	4.03	24.19
FTP UNKNOWN	11	54.55	0.00	0.00	45.45
HTTP FILETRANSFER	34676	97.56	1.52	0.80	0.12
HTTP UNKNOWN	1583731	99.13	0.00	0.01	0.86
HTTP WEBBROWSING	2355298	86.06	13.30	0.03	0.61
HTTP WEBRADIO	326	92.64	0.31	7.06	0.00
HTTPS UNKNOWN	748695	91.94	0.00	0.41	7.66
NETBIOS UNKNOWN	13133	99.97	0.00	0.00	0.03
NTP INTERACTIVE	58375	99.80	0.00	0.01	0.19
RDP UNKNOWN	12	8.33	0.00	83.33	8.33
RTMP STREAMING	497	85.92	0.00	4.83	9.26
SKYPE CONVERSATION	2480	89.35	1.09	3.95	5.60
SKYPE UNKNOWN	483562	98.51	0.00	0.95	0.55
For all flows together	17190269	97.33	1.83	0.09	0.75
Average from all classes		86.61	1.05	6.03	6.31

7.2 Building and Testing of the Classifier

The method of building and testing of the classifier for the *application*, *behavior*, and *content* levels was the same as for the dataset with full packet payloads, which was previously described in Section 6.4.

7.3 Results

The classification results are shown and discussed separately for each classification module.

7.3.1 Application and Behavior Classification Module

The results obtained by this module are shown in Table 7.2. We present the number of flows in each traffic category (based on the ground truth) together with the percentage of flows, which were classified correctly, wrong, or as *UNKNOWN*. The category notation is the same as for the dataset with full packet payloads, which was previously described in Section 6.4.

As observed, the total error rate for all the flows was 0.09 %, while 0.75 % of flows remained unknown. If we look at the average from all the traffic classes, the error rate was bigger (6.03 %) and 6.31 % of flows remained unknown. As shown in Table 7.2, the per-class error rate was higher due to several classes, which contained only a few flows. An example can be *RDP UNKNOWN*, which contained only 12 flows, which were mostly misclassified. However, the misclassification was not only made due to poor training on a limited number of flows; in our dataset we could see that the possessed by us *RDP UNKNOWN* flows were just connection requests, which were probably rejected from the server, so no real RDP connection was established.

Table 7.3: Classification by the *Content Classification Module* of VBS Dataset

Content	Flows	Correct [%]	Wrong [%]	Unknown [%]
AUDIO	1765	80.11	6.01	13.88
MULTIMEDIA	497	86.12	0.20	13.68
VIDEO	23511	87.27	0.60	12.13
WWW	2355298	86.06	0.00	13.94
UNKNOWN	20952650	98.94	1.06	0.00
All flows	23333721	97.62	0.96	1.42
Average from classes		87.70	1.58	10.73

Additionally, we tried to identify 6 143 452 flows, which ground truth was given as *UNKNOWN UNKNOWN*: 29.41 % of them were classified as *DNS INTERACTIVE*, 2.42 % as *BITTORRENT UNKNOWN*, 1.97 % as other classes, while 66.20 % remained as *UNKNOWN UNKNOWN*.

7.3.2 Content Classification Module

The accuracy of the classification of the particular types of contents is shown in Table 7.3. As presented, we obtained very low classification error (below 1 %) within the flows from the *MULTIMEDIA*, *VIDEO*, and *WWW* classes, and higher classification error of *AUDIO* flows (6.01 %). The rate of the unclassified flows fluctuated around 13 % for all the classes. In our case, the ground truth for the flows labeled as *UNKNOWN* was established as non-*AUDIO*, non-*MULTIMEDIA*, non-*VIDEO*, and non-*WWW*. We obtained only 1.06 % of errors for this class, which means that the same IP addresses were used both for the flows transmitting files from the *AUDIO*, *MULTIMEDIA*, *VIDEO*, and *WWW* classes, and for flows transmitting other kinds of files. In total, we classified properly 97.62 % of flows, 0.96 % of flows were classified wrong, and 1.42 % remained unknown.

Chapter 8

Discussion and Comparison of the Results

The results show that our classifier is able to identify the traffic on multiple different levels with high accuracy. What is unique for tools relying on Machine Learning capabilities, our classifier can properly handle traffic from non-recognized applications, marking it as *UNKNOWN*. We also compared the results of classification by our classifier to the results given by various DPI tools.

PACE. This DPI tool from *ipoque* [49] provides very accurate results on the *application* level. We assessed its accuracy as 93.78 % on our dataset with full payloads. However, some of the results are given on *content* level instead, as *FLASH*, *QUICKTIME*, or *WINDOWSMEDIA*. In these cases, we do not really know what is the application. For example, *FLASH* content can be transmitted by both *HTTP* or *RTMP* application protocols. Furthermore, the *FLASH* content can be streamed (as in *RTMP*) or just downloaded to the user's computer, and then saved to a permanent file, or played by the browser (as in the case of YouTube videos, which use *HTTP*). Furthermore, we do not have knowledge about the service provider names.

OpenDPI. This DPI tool was an open-source fork of PACE, with removed support for encrypted protocols and optimization functions. Its accuracy on the *application* level in our case was 54.19 %. Therefore, the range of the values provided by OpenDPI is almost the same as returned by PACE.

NDPI. This DPI tool is an open-source fork of OpenDPI, which was extended to support some of the protocols, which were removed in OpenDPI. Its accuracy on the *application* level was in our case 55.76 %. Furthermore, it is able to provide the classification on the *service provider* level, as *facebook*, *google*, or *twitter*. However, the final output of the classification is mixed on different levels. For some flows we only obtain the application name (as *dns* or *bittorrent*), for some we only obtain the content (as *flash*), and for some we only obtain the service provider name (as *facebook*). Based on the application name we cannot estimate what is the service provider or the content, and vice versa.

Libprotoident. This tool presents a method called Lightweight Packet Inspection (LPI) [50], as it inspects only first four Bytes of payload in each direction. Therefore, it can be used in many places, where the collected payload must be truncated due to privacy reasons or legal issues. Its accuracy on the *application* level in our case was 94.04 %, which means that the tool achieved the highest accuracy among all of the tested DPI tools. The output from the classifier seems to be also structured in an interesting way, since for many application protocols it gives also information about the transport-layer protocol (as *DNS_TCP*, *BitTorrent_UDP*, or

Unknown_UDP), which is also unique among all the tested DPI tools. However, many flows obtain the classification only on the *content* level (as *Flash_Player*), or the *service provider* level (as *YahooError*).

NBAR. Network-Based Application Recognition (NBAR) was added to many Cisco devices as a feature, which was supposed to work together with other functions of these devices, as traffic filtering or shaping. This tool combines different techniques, as port-based classification and DPI. Its accuracy on the *application* level was in our case 25.18 %. This tool provides a very consistent output, as all the results are given on the *application* level. Despite that, the classification accuracy of other tools tested by us was higher.

L7-Filter. L7-filter [51] is an open-source combination of the DPI engine and rules, which can be downloaded separately and applied to the traffic in various configurations, depending on their status and order. Since the last version of the rules is from 2009, the accuracy on the application level was in our case 16.19–30.21 % depending on the configuration. As we observed, the results were returned always on the application level.

UPC Classifier. Another interesting classification approach is shown by authors of a tool developed at UPC BarcelonaTech [52]. The authors performed extensive work regarding traffic classification by various MLAs (including C5.0). They succeeded in comparing various classification techniques in distinguishing of 14 different application classes, and they achieved accuracy of 88-97 %. The final classification tool is based on C5.0 and it uses an automatic retraining system. It is shown to provide in the particular network results, which are comparable to the results achieved by PACE. The ground truth for the training data is established based on DPI techniques: at first, the class of a flow is tried to be obtained by PACE. If PACE cannot provide the result, L7-filter and NDPI are used. Only the cases for which the ground truth is obtained are provided as the training data, so the classifier does not provide any *UNKNOWN* results – all the cases are assigned to the most probable class. Besides that, the classifier also uses some time-based parameters, as inter-arrival times of packets or duration of the flow, so the results can depend on the conditions in the network. The format of the results is a mix of the formats of results provided by PACE, L7-filter, and NDPI.

In our study, we tried to cover the biggest possible extent of applications and their behaviors. It has, however, many limitations:

- Our classifier is able to recognize only 16 applications and 14 different behaviors in total. Deep Packet Inspection tools as well as automatically trained classifiers are able to recognize thousands of different applications. The construction of the multilevel classifier requires us to manually create rules for each application (or group of applications) and for each of its behaviors. Therefore, we could include in our study only these applications, from which we collected sufficient amount of traffic. Nevertheless, our classification covers nearly 90 % of the total data volume in the dataset with full payloads and 85 % of the data volume in the second dataset.

- Our datasets contain limited number of applications. The first dataset, which includes full payloads, was created by us using 3 virtual machines and running the selected applications. The second dataset contains the data obtained from 54 users from Denmark and Poland over 1.5 years. However, the users can be considered as not being enough representative – half of them are students using computers in a school, and the majority of the rest are highly qualified network users.

- The amount and the character of the flows for which we established the ground truth also has an impact on the overall accuracy. As shown, we were not able to establish the ground truth for around 60 % of flows in the first dataset and for 26 % of flows in the second dataset. These flows in majority contain less than 10 packets, usually 2–4, so the socket name is not noticed by VBS. Therefore, our results should be considered as results of classification of longer flows.

- For the second dataset (without packet payloads), we were not able to evaluate the classifier on the *Ethernet*, *IP protocol*, and *service provider* levels. Although the classifier should be always accurate on the *Ethernet* and *IP protocol* levels, the accuracy of the classification on the *service provider* level is a big question mark.
- Considering a big variety of different types of contents and service providers, we did not study how much training is needed for the classifier to provide high accuracy in a real environment. We neither know, what is the accuracy of the classifier trained in one environment and working in another one.
- Constructing the rules used to establish the ground truth for the selected applications and their behaviors is a time-consuming task. The time spent on that can be compared with the time needed to construct DPI rules.

Chapter 9

Conclusions

This report introduces a novel hybrid method for traffic classification and accounting, which was created to overcome the drawbacks of already existing methods and tools. The classification is performed on six levels: *Ethernet*, *IP protocol*, *application*, *behavior*, *content*, and *service provider*.

The system created based on the method was tested on two datasets and it was shown that it provides accurate results on all the levels. The classification error did not exceed 1.5 % on any level during the evaluation on both datasets. Since the method does not require to inspect the application-layer packet payloads, the classification is fast and lightweight. So, it can be performed even in high-speed networks. Moreover, the process does not interfere with users' privacy issues. Due to the consistent classification of each flow on all the levels, the accounted traffic can be used to generate many useful reports. The reports can be used by ISPs to improve their services, lower costs, and attract new customers.

Our open-source implementation of the system in Java can be downloaded from Sourceforge [11]. The future work will concentrate on introducing support for new applications and their behaviors.

References

[1] Riyad Alshammari and A. Nur Zincir-Heywood. Machine Learning based encrypted traffic classification: identifying SSH and Skype. In *Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA 2009)*, pages 1–8. IEEE, Ottawa, Ontario, Canada, July 2009. DOI: 10.1109/CISDA.2009.5356534.

[2] Sven Ubik and Petr Žejdl. Evaluating application-layer classification using a Machine Learning technique over different high speed networks. In *Proceedings of the Fifth International Conference on Systems and Networks Communications (ICSNC)*, pages 387–391. IEEE, Nice, France, August 2010. DOI: 10.1109/ICSNC.2010.66.

[3] Jun Li, Shunyi Zhang, Yanqing Lu, and Junrong Yan. Real-time P2P traffic identification. In *Proceedings of the IEEE Global Telecommunications Conference (IEEE GLOBECOM 2008)*, pages 1–5. IEEE, New Orleans, Louisiana, USA, December 2008. DOI: 10.1109/GLOCOM.2008.ECP.475.

[4] Ying Zhang, Hongbo Wang, and Shiduan Cheng. A Method for Real-Time Peer-to-Peer Traffic Classification Based on C4.5. In *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT)*, pages 1192–1195. IEEE, Nanjing, China, November 2010. DOI: 10.1109/ICCT.2010.5689126.

[5] Jing Cai, Zhibin Zhang, and Xinbo Song. An analysis of UDP traffic classification. In *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT)*, pages 116–119. IEEE, Nanjing, China, November 2010. DOI: 10.1109/ICCT.2010.5689203.

[6] Riyad Alshammari and A. Nur Zincir-Heywood. Unveiling Skype encrypted tunnels using GP. In *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, Barcelona, Spain, July 2010. DOI: 10.1109/CEC.2010.5586288.

[7] L7-filter Supported Protocols, 2012. [Online]. Available: `http://l7-filter.sourceforge.net/protocols`.

[8] Li Jun, Zhang Shunyi, Lu Yanqing, and Zhang Zailong. Internet Traffic Classification Using Machine Learning. In *Proceedings of the Second International Conference on Communications and Networking in China (CHINACOM '07)*, pages 239–243. IEEE, Shanghai, China, August 2007. DOI: 10.1109/CHINACOM.2007.4469372.

[9] Yongli Ma, Zongjue Qian, Guochu Shou, and Yihong Hu. Study of Information Network Traffic Identification Based on C4.5 Algorithm. In *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*, pages 1–5. IEEE, Dalian, China, October 2008. DOI: 10.1109/WiCom.2008.2678.

[10] Wei Li and Andrew W. Moore. A Machine Learning Approach for Efficient Traffic Classification. In *Proceedings of the Fifteenth IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'07)*, pages 310–317. IEEE, Istanbul, Turkey, October 2007. DOI: 10.1109/MASCOTS.2007.2.

[11] Volunteer-Based System for Research on the Internet, 2012. [Online]. Available: http://vbsi.sourceforge.net/.

[12] Jason But, Philip Branch, and Tung Le. Rapid identification of BitTorrent Traffic. In *Proceedings of the 35th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 536–543. IEEE, Denver, Colorado, USA, October 2010. DOI: 10.1109/LCN.2010.5735770.

[13] Kei Takeshita, Takeshi Kurosawa, Masayuki Tsujino, Motoi Iwashita, Masatsugu Ichino, and Naohisa Komatsu. Evaluation of HTTP video classification method using flow group information. In *Proceedings of the 14th International Telecommunications Network Strategy and Planning Symposium (NETWORKS)*, pages 1–6. IEEE, Warsaw, Poland, September 2010. DOI: 10.1109/NETWKS.2010.5624929.

[14] Samruay Kaoprakhon and Vasaka Visoottiviseth. Classification of audio and video traffic over HTTP protocol. In *Proceedings of the 9th International Symposium on Communications and Information Technology (ISCIT 2009)*, pages 1534–1539. IEEE, Icheon, Korea, September 2009. DOI: 10.1109/ISCIT.2009.5341026.

[15] CAIDA Internet Data – Passive Data Sources, 2012. [Online]. Available: http://www.caida.org/data/passive/.

[16] CAIDA Internet Data – Internet Measurement Data Catalog (IMDC), 2012. [Online]. Available: http://www.caida.org/projects/trends/imdc/.

[17] Colleen Shannon, David Moore, Ken Keys, Marina Fomenkov, Bradley Huffaker, and Kimberly C. Claffy. The internet measurement data catalog. *ACM SIGCOMM Computer Communication Review*, 35(5):97–100, October 2005. DOI: 10.1145/1096536.1096552.

[18] MAWI Working Group Traffic Archive, 2013. [Online]. Available: http://mawi.wide.ad.jp/mawi/.

[19] Kensuke Fukuda. Difficulties of identifying application type in backbone traffic. In *2010 International Conference on Network and Service Management (CNSM)*, pages 358–361. IEEE, Niagara Falls, Ontario, Canada, October 2010. DOI: 10.1109/CNSM.2010.5691234.

[20] CRAWDAD, 2013. [Online]. Available: http://crawdad.cs.dartmouth.edu/.

[21] Xiaoguo Zhang and Wei Ding. Comparative Research on Internet Flows Characteristics. In *2012 Third International Conference on Networking and Distributed Computing (ICNDC)*, pages 114–118. IEEE, Hangzhou, China, October 2012. DOI: 10.1109/ICNDC.2012.35.

[22] CERNET data traces, 2012. [Online]. Available: http://ntds.njnet.edu.cn/data/index.php.

[23] WITS: Waikato Internet Traffic Storage, 2013. [Online]. Available: http://www.wand.net.nz/wits/.

[24] Tomasz Bujlow, Kartheepan Balachandran, Sara Ligaard Nørgaard Hald, Tahir Riaz, and Jens Myrup Pedersen. Volunteer-Based System for research on the Internet traffic. *TELFOR Journal*, 4(1):2–7, September 2012.

[25] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. A method for classification of network traffic based on C5.0 Machine Learning Algorithm. In *Proceedings of ICNC'12: 2012 International Conference on Computing, Networking and Communications (ICNC): Workshop on Computing, Networking and Communications*, pages 244–248. IEEE, Maui, Hawaii, USA, February 2012. DOI: 10.1109/ICCNC.2012.6167418.

[26] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. A method for Assessing Quality of Service in Broadband Networks. In *Proceedings of the 14th International Conference on Advanced Communication Technology (ICACT)*, pages 826–831. IEEE, Phoenix Park, PyeongChang, Korea, February 2012.

[27] Tomasz Bujlow, Sara Ligaard Hald, M. Tahir Riaz, and Jens Myrup Pedersen. A Method for Evaluation of Quality of Service in Computer Networks. *ICACT Transactions on the Advanced Communications Technology (TACT)*, 1(1):17–25, July 2012.

[28] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. Classification of HTTP traffic based on C5.0 Machine Learning Algorithm. In *Proceedings of the Fourth IEEE International Workshop on Performance Evaluation of Communications in Distributed Systems and Web-based Service Architectures (PEDISWESA 2012)*, pages 882–887. IEEE, Cappadocia, Turkey, July 2012. DOI: 10.1109/ISCC.2012.6249413.

[29] Jens Myrup Pedersen and Tomasz Bujlow. Obtaining Internet Flow Statistics by Volunteer-Based System. In *Fourth International Conference on Image Processing & Communications (IP&C 2012), Image Processing & Communications Challenges 4, AISC 184*, pages 261–268. Springer Berlin Heidelberg, Bydgoszcz, Poland, September 2012. DOI: 10.1007/978-3-642-32384-3_32.

[30] Tomasz Bujlow and Jens Myrup Pedersen. Obtaining Application-based and Content-based Internet Traffic Statistics. In *Proceedings of the 6th International Conference on Signal Processing and Communication Systems (ICSPCS'12)*, pages 1–10. IEEE, Gold Coast, Queensland, Australia, December 2012. DOI: 10.1109/ICSPCS.2012.6507984.

[31] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. Comparison of Deep Packet Inspection (DPI) Tools for Traffic Classification. Technical report, Department of Computer Architecture (DAC), Universitat Politècnica de Catalunya (UPC), June 2013. Accessible: `https://www.ac.upc.edu/app/research-reports/html/research_center_index-CBA-2013,en.html`.

[32] Valentín Carela-Español, Tomasz Bujlow, and Pere Barlet-Ros. Is our ground-truth for traffic classification reliable? In *To appear in the Passive and Active Measurement Conference (PAM 2014)*, pages ?–? DBLP, Los Angeles, USA, March 2014.

[33] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. Extended Independent Comparison of Popular Deep Packet Inspection (DPI) Tools for Traffic Classification. Technical report, Department of Computer Architecture (DAC), Universitat Politècnica de Catalunya (UPC), January 2014. Accessible: `https://www.ac.upc.edu/app/research-reports/html/research_center_index-CBA-2014,en.html`.

[34] Information on See5/C5.0 – RuleQuest Research Data Mining Tools, 2011. [Online]. Available: `http://www.rulequest.com/see5-info.html`.

[35] Zhou Xusheng and Zhou Yu. Application Of Clustering Algorithms In IP Traffic Classification. In *Proceedings of the WRI Global Congress on Intelligent Systems (GCIS'09)*, volume 2, pages 399–403. IEEE, Xiamen, China, May 2009. DOI: 10.1109/GCIS.2009.139.

[36] Liu Bin and Tu Hao. An Application Traffic Classification Method Based on Semi-Supervised Clustering. In *Proceedings of the 2nd International Symposium on Information Engineering and Electronic Commerce (IEEC)*, pages 1–4. IEEE, Ternopil, Ukraine, July 2010. DOI: 10.1109/IEEC.2010.5533239.

[37] Xiang Li, Feng Qi, Li Kun Yu, and Xue Song Qiu. High accurate Internet traffic classification based on co-training semi-supervised clustering. In *Proceedings of the International Conference on Advanced Intelligence and Awareness Internet (AIAI 2010)*, pages 193–197. IET, Beijing, China, October 2010. DOI: 10.1049/cp.2010.0751.

[38] Jing Yuan, Zhu Li, and Ruixi Yuan. Information Entropy Based Clustering Method for Unsupervised Internet Traffic Classification. In *Proceedings of the IEEE International Conference on Communications (ICC'08)*, pages 1588–1592. IEEE, Beijing, China, May 2008. DOI: 10.1109/ICC.2008.307.

[39] Thuy T. T. Nguyen and Grenville Armitage. Clustering to Assist Supervised Machine Learning for Real-Time IP Traffic Classification. In *Proceedings of the IEEE International Conference on Communications (ICC'08)*, pages 5857–5862. IEEE, Beijing, China, May 2008. DOI: 10.1109/ICC.2008.1095.

[40] Caihong Yang and Benxiong Huang. Traffic classification using an improved clustering algorithm. In *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS 2008)*, pages 515–518. IEEE, Xiamen, Fujian, China, May 2008. DOI: 10.1109/ICCCAS.2008.4657826.

[41] Liu Yingqiu, Li Wei, and Li Yunchun. Network Traffic Classification Using K-means Clustering. In *Proceedings of the Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, pages 360–365. IEEE, Iowa City, Iowa, USA, August 2007. DOI: 10.1109/IMSCCS.2007.52.

[42] Pedro Casas, Johan Mazel, and Philippe Owezarski. On the use of Sub-Space Clustering & Evidence Accumulation for traffic analysis & classification. In *Proceedings of the 7th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 1016–1021. IEEE, Istanbul, Turkey, July 2011. DOI: 10.1109/IWCMC.2011.5982680.

[43] Narendra Sharma, Aman Bajpai, and Ratnesh Litoriya. Comparison the various clustering algorithms of weka tools. *International Journal of Emerging Technology and Advanced Engineering*, 2(5):73–80, May 2012.

[44] IEEE Public EtherType list, 2013. [Online]. Available: `http://standards.ieee.org/develop/regauth/ethertype/eth.txt`.

[45] Protocol Numbers, 2013. [Online]. Available: `http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xml`.

[46] Tomasz Bujlow, Kartheepan Balachandran, Tahir Riaz, and Jens Myrup Pedersen. Volunteer-Based System for classification of traffic in computer networks. In *Proceedings of the 19th Telecommunications Forum TELFOR 2011*, pages 210–213. IEEE, Belgrade, Serbia, November 2011. DOI: 10.1109/TELFOR.2011.6143528.

[47] Kartheepan Balachandran and Jacob Honoré Broberg. Volunteer-Based Distributed Traffic Data Collection System. Master's thesis, Aalborg University, Department of Electronic Systems, Denmark, June 2010.

[48] De Sensi, Daniele and Danelutto, Marco and Deri, Luca. Dpi over commodity hardware: implementation of a scalable framework using fastflow. Master's thesis, Università di Pisa, Italy, 2012. Accessible: `http://etd.adm.unipi.it/t/etd-02042013-101033/`.

[49] PACE | Network Analysis with Layer-7 Deep Packet Inspection, 2013. [Online]. Available: `http://www.ipoque.com/en/products/pace`.

[50] Shane Alcock and Richard Nelson. Libprotoident: Traffic Classification Using Lightweight Packet Inspection. Technical report, University of Waikato, August 2012. Accessible: `http://www.wand.net.nz/publications/lpireport`.

[51] Application Layer Packet Classifier for Linux, 2009. [Online]. Available: http://l7-filter.sourceforge.net/.

[52] Oriol Mula-Valls. A practical retraining mechanism for network traffic classification in operational environments. Master's thesis, Computer Architecture, Networks and Systems, Universitat Politècnica de Catalunya, Spain, June 2011. Accessible: https://www.ac.upc.edu/app/research-reports/html/research_center_index-CBA-2011,en.html.

Appendix A

C5.0 Classification Attributes

The classification by C5.0 relies on the following 50 attributes:

- Identifier of the flow, used to match the classification results with the original database records
- Local and remote port numbers
- Transport protocol name
- Number of packets and bytes carried by the flow
- Percent of packets and bytes, which go in the inbound direction
- Average, maximum, minimum, first quartile, median, third quartile, and standard deviation of total, inbound, and outbound packet size
- Percent of total, inbound, and outbound packets, which carry any data
- Percent of total, small ($< 70\,\mathrm{B}$), and big ($> 1320\,\mathrm{B}$) packets carrying data, which are going into the inbound direction
- Percent of total, inbound, and outbound packets carrying data, which are small ($< 70\,\mathrm{B}$) and big ($> 1320\,\mathrm{B}$)
- Percent of total, inbound, and outbound packets, which have ACK and PSH flag set
- Percent of packets containing ACK, and PSH flag, which are going in the inbound direction
- Class name, created as: APPLICATION_BEHAVIOR

Appendix B

Rules Used to Establish the Ground Truth on the *Application* and *Behavior* Levels

The rules used to establish the ground truth on the *application* and *behavior* levels for the particular applications are given below.

HTTP. Used for HTTP traffic. Any packet belonging to that flow must contain an HTTP header.

WEBRADIO: Used for Internet radio streams. More than 50 % of the inbound packets of that flow must have PSH flag set. Additionally, the flow must consists of at least 1000 packets, and it must carry content of a type of *audio/mpeg*, *audio/aac*, *audio/aacp*, or *audio/mp4*. This behavior concerns only live audio streams; audio files downloaded and played by the browser are labeled by *FILETRANSFER* behavior.

FILETRANSFER: Used for transfer of big files. At least one file transmitted by this HTTP flow must be bigger than 1 MB. This behavior concerns any transfer of a big file by HTTP protocol. YouTube video files represent this behavior as well, as they are normally downloaded to the users' computers by HTTP, and then they are played (usually chunk by chunk using *progressive download* technique).

WEBBROWSING: Used for interactive browsing of web sites. The name of the application belonging to the flow must be *chrome*, *firefox*, or *iexplore*. Additionally, the flow must not transport any file bigger than 256 kB with exceptions of images of types *image/jpeg*, *image/png*, *image/gif*, *image/bmp*, *image/jpg*, *image/webp*, which can be transported of any size.

UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

HTTPS. Used for HTTPS traffic. The name of the application belonging to the flow must be *chrome*, *firefox*, or *iexplore*. Additionally, the flow must not carry any packet with an HTTP header, and the remote port number must be 443.

UNKNOWN: All the flows present this behavior.

AMERICASARMY. Used for America's Army first-player shooter game. The name of the application belonging to the flow must be *aa3game*.

GAMESESSION: Used for main flows generated by a first-player shooter game called America's Army. They use *UDP* transport protocol and there must be at least 200 packets in the flow.

UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

BITTORRENT. Used for BitTorrent traffic. The name of the application belonging to the flow must be *utorrent*, *bittorrent*, *frostwire*, or *azureus*.

FILETRANSFER: Used for flows, which contain downloads and uploads of files. The flow must amount for at least 256 kB. More than 90 % of bytes, for which the flow amounts, must be transferred in one of the directions.

UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

DHCP. Used for DHCP traffic. The name of the application belonging to the flow must be *svchost* or *dhclient*, the local port number must be 68, remote port number 67, and the flow must use *UDP* transport protocol.

INTERACTIVE: All the flows present this behavior.

DNS. Used for DNS traffic. The name of the application belonging to the flow must be *svchost* or *dnsmasq* and the remote port number must be 53.

INTERACTIVE: All the flows present this behavior.

EDONKEY. Used for Edonkey traffic. The name of the application belonging to the flow must be *amule* or *emule*.

FILETRANSFER: Used for flows, which present downloads and uploads of files. The flow must amount for at least 180 kB. More than 90 % of bytes, for which the flow amounts, must be transferred in one of the directions.

UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

FTP. Used for FTP traffic. The name of the application belonging to the flow must be *filezilla*, *smartftp*, *ftpte*, or *winscp*.

CONTROL: Used for control flows. The flow must use remote port number 21 and not more than 90 % of bytes, for which the flow amounts, must be transferred in one of the directions.

FILETRANSFER: Used for transfers of big files. More than 90 % of bytes, for which the flow amounts, must be transferred in one of the directions.

UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

NETBIOS. Used for NETBIOS traffic. The name of the application belonging to the flow must be *system* and the local and remote port numbers must be 137.

UNKNOWN: All the flows present this behavior.

NTP. Used for NTP traffic. The name of the application belonging to the flow must be *ntpd* or *svchost*, the local and remote port numbers must be 123, and the flow must use *UDP* transport protocol.

INTERACTIVE: All the flows present this behavior.

RDP. Used for Remote Desktop traffic. The name of the application belonging to the flow must be *svchost* or *xrdp* and the local port number must be 3389.

UNKNOWN: All the flows present this behavior.

RTMP. Used for live video and audio transmissions, which use RTMP protocol. Two kinds of flows are included in this group:

- The name of the application belonging to the flow must be *libgcflashpla**, *plugin-contai**, *chrome*, *firefox*, or *iexplore*. Additionally, the flow must contain at least 5001 packets and the remote port number must be 1935.

- The name of the application belonging to the flow must be *libgcflashpla** or *plugin-contai**. Additionally, the flow must contain at least 5001 packets and the remote port number must be 80.

STREAMING: All the flows present this behavior.

SKYPE. Used for Skype traffic. The name of the application belonging to the flow must be *skype*.
CONVERSATION: Used for voice and video conversations, file transfers. The number of packets contained by this flow must be at least 700.
UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

SSH. Used for SSH traffic. The name of the application belonging to the flow must be *sshd* or *sshd:*.
UNKNOWN: All the flows present this behavior.

DOWNLOADER. Used for traffic from various file downloaders. The name of the application belonging to the flow must be *dropbox*, *chomikbox*, or *steam*.
FILETRANSFER: Used for transfers of big files. The flow must amount for at least 5 MB.
UNKNOWN: Assigned to these flows, which do not comply to any of the previously described behaviors.

UNKNOWN. Used for all the flows, which do not fit to the above specified classes.
UNKNOWN: All the flows present this behavior.

Appendix C

Rules Used to Detect the Suspicious Unknown Traffic

We created the following rules for the suspicious traffic, which will be partially excluded from the training data for C5.0:

a) Suspicious to be DHCP traffic:

   Flows classified as *UNKNOWN UNKNOWN*, without any application name collected from system sockets. No packet contained by the flow can contain an HTTP header. Additionally, the local port number must be 68, remote port number 67, and the flow must use UDP transport protocol.

b) Suspicious to be DNS traffic:

   Flows classified as *UNKNOWN UNKNOWN*, without any application name collected from system sockets. No packet contained by the flow can contain an HTTP header. Additionally, the remote port number must be 53.

c) Suspicious to be NTP traffic:

   Flows classified as *UNKNOWN UNKNOWN*, without any application name collected from system sockets. No packet contained by the flow can contain an HTTP header. Additionally, the local and remote port numbers must be 123 and the flow must use UDP transport protocol.

Appendix D

Service Providers Used During the Evaluation of the Classifier on the Dataset with Full Payloads

Based on the collected data, we decided to select the following 14 service providers to use in the experiment:

a) DOUBLECLICK.NET: for flows, where the *URL* field in the HTTP header contains the pattern: *doubleclick.net* or *2mdn.net*

b) YOUTUBE: for flows, where the *URL* field in the HTTP header contains the pattern: *youtube.*, *ytimg.com*, or *youtube-nocookie.*

c) FACEBOOK: for flows, where the *URL* field in the HTTP header contains the pattern: *facebook.* or *fbcdn.net*

d) YAHOO: for flows, where the *URL* field in the HTTP header contains the pattern: *yahoo.*, *yimg.com*, *yahooapis.*, *yieldmanager.*, or *bluelithium.com*

e) ORACLE: for flows, where the *URL* field in the HTTP header contains the pattern: *oracle.*

f) WIKIPEDIA: for flows, where the *URL* field in the HTTP header contains the pattern: *wikipedia.*, *wikimedia.*, *mediawiki.*, or *wikimediafoundation.*

g) JUSTIN.TV: for flows, where the *URL* field in the HTTP header contains the pattern: *justin.tv*, *justintvlivefs.*, or *jtvnw.net*

h) GOOGLE: for flows, where the *URL* field in the HTTP header contains the pattern: *google.*, *googleads.*, *google-analytics.*, *googlesyndication.*, *googleusercontent.*, *googleadservices.*, *googletagservices.*, or *gstatic.com*

i) UBUNTU.COM: for flows, where the *URL* field in the HTTP header contains the pattern: *ubuntu.com*

j) TWITTER: for flows, where the *URL* field in the HTTP header contains the pattern: *twitter.* or *twimg.com*

k) MICROSOFT.COM: for flows, where the *URL* field in the HTTP header contains the pattern: *microsoft.com*, *windowsupdate.com*, or *atdmt.com*

l) KING-ONLINE.RU: for flows, where the *URL* field in the HTTP header contains the pattern: *king-online.ru*

m) SCORECARDRESEARCH.COM: for flows, where the *URL* field in the HTTP header contains the pattern: *scorecardresearch.com*

n) TRIBALFUSION.COM: for flows, where the *URL* field in the HTTP header contains the pattern: *tribalfusion.com*

A Practical Method for Multilevel Classification and Accounting of Traffic in Computer Networks

TOMASZ BUJLOW, JENS MYRUP PEDERSEN

Existing tools for traffic classification are shown to be incapable of identifying the traffic in a consistent manner. For some flows only the application is identified, for others only the content, for yet others only the service provider. Furthermore, Deep Packet Inspection is characterized by extensive needs for resources and privacy or legal concerns. Techniques based on Machine Learning Algorithms require good quality training data, which are difficult to obtain. They usually cannot properly deal with other types of traffic, than they are trained to work with, and they are unable to detect the content carried by the flow, or the service provider.

To overcome the drawbacks of already existing methods, we developed a novel hybrid method to provide accurate identification of computer network traffic on six levels: Ethernet, IP protocol, application, behavior, content, and service provider. Our system built based on the method provides also traffic accounting and it was tested on 2 datasets. We have shown that our system gives a consistent, accurate output on all the levels. We also showed that the results provided by our system on the application level outperformed the results obtained from the most commonly used DPI tools.

2014
Aalborg University
Department of Electronic Systems
Networking & Security
www.es.aau.dk/netsec